AUTONOMOUS DIAL-A-RIDE TRANSIT SYSTEM : CONSTRUCTION METHODS

A Thesis Submitted

in Partial Fulfillment of the Requirements for the Degree of

MASTER OF TECHNOLOGY

by

KAPIL JAIN

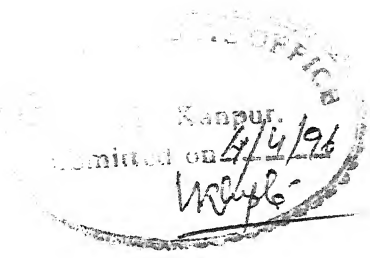
to the

DEPARTMENT OF INDUSTRIAL AND MANAGEMENT ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY, KANPUR

APRIL, 1996.

CERTIFICATE



This is to certify that the work contained in this thesis entitled "Autonomous Dial-a-Ride Transit System : Construction Methods" by Kapil Jain (Roll No. 9411409) has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.

(**Ravindra K. Ahuja**)
Professor
Industrial and Management Department
Indian Institute of Technology
Kanpur - 208016

April, 1996.

ACKNOWLEDGEMENTS

I take this opportunity to convey my sincere thanks to Dr. R. K. Ahuja for his guidance and encouragement. He introduced me to this very interesting class of Dial-A-Ride problems. He was a constant source of inspiration throughout my stay over here. Working with him on these very interesting problems was a perfect way to finish the studies.

I would also like to thank Deepak for his constant support to me while working on the thesis. The stay here become more pleasant with the presence of Negi, Shobhit Parvesh and whole IME gang.

I am also thankful to the staff members of IME department for their cooperation.

April Kapil Jain

ABSTRACT

Autonomous Dial-A-Ride Transit (ADART) system is a mass transit system that would serve areas of high travel demand. It is a shared taxi system, thus provides cheaper alternate compared to normal taxi service. But it is costlier than the normal bus service.

In this thesis we have developed computerized algorithms to solve advance request version of ADART system. Advance requests are those requests that have been received before the start of a day. We have used construction based heuristic approaches to generate feasible schedules.

We have further developed randomization approaches to generate improved schedules. Finally, we have developed a taxi simulation model. Computational experience on the comparative performance of taxi simulation and our approaches is presented for 1000 requests and 60 simultaneously operating vehicles.

TABLE OF CONTENTS

CHAPTER 1 INTRODUCTION

1.1 INTRODUCTION TO AUTONOMOUS DIAL-A-RIDE TRANSIT SYSTEM

Autonomous dial-a-ride Transit (ADART) system is a mass transit system which would serve areas of high travel demand. It is a modernized version of many-to-few dial-a-ride transportation system. It employs fully automated order-entry and dispatching systems that reside on board the vehicle. An ADART vehicle fleet efficiently serves travel demand in a large geographical area without need of centralized supervision.

Each vehicle's on-board computer receives a customer's trip request, inserts this request into the vehicle's schedule, and plans an optimal route to accomplish the schedule. The computer controls the vehicle's route by continuously informing the driver of the next required change of direction. Furthermore, if advantageous to do so, the computer may pass responsibility for a trip request off to another vehicle-computer better positioned to provide the service.

Due to its distributed command-and-control, ADART promises to provide better service than conventional dial-a-ride and at lower operating cost. It increases driver productivity and reduces total manpower. It can readily increase or reduce its fleet size at a fixed operating cost per vehicle. Redeploying vehicles from one service area to another is almost free. Furthermore, in contrast with conventional dial-a-ride, the greater the

demand, the better the ADART system performs. The result could be a transportation service for low density areas that is attractive to customer and supplier alike.

This thesis is devoted to the development of a set of tools that can help to operate such vehicles in an efficient manner. The focus of our work is on designing computerized algorithms which will decide the routes of vehicles. Section 1.2 discusses various modeling issues related to the vehicle routing and scheduling. In Section 1.3 we explain the purpose of this thesis and present an outline of the work.

1.2 Modeling Issues In Vehicle Routing And Scheduling Problems

Modeling of any problem is a very important issue. In this section we will discuss various issues of vehicle routing in general. Firstly, we will describe the general characteristics of vehicle routing and scheduling problem. Then we will describe various constraints that may one has to confront regarding the generated routes. Since these problems are multi criterion problems, thus the choice of objective function is plays vary important role in the performance of ADART. Lastly, we have discussed the factors that should be considered in the objective function.

(A) General Characteristics of Routing Problems.

NATURE OF DEMAND

- Pure pickups and pure deliveries
- Pickups (deliveries) with back haul options
- Single or multiple commodities
- Must serve all demand?
- Common carrier option
- Priorities for customer

INFORMATION ON DEMAND

- All demand known in advance?
- Many repeat demands?

- Fixed frequencies for visits?
- Uncertain demands?
- Real-time inflow of demands

VEHICLE FLEET

- Homogeneous fleet or multiple vehicle types
- Weight and capacity restrictions
- Loading restrictions/equipment
- Vehicle type/site dependencies
- Vehicle type/commodity compatibility
- Fixed or variable fleet size
- Fleet based at single depot or multiple terminals

CREW REQUIREMENTS

- Pay structure
- Length of workday
- Minimum and maximum on duty times
- Overtime option
- Fixed or variable number of drivers
- Driver start times and locations
- Lunch or other breaks
- Multiple-day trips allowed

SCHEDULING REQUIRIEMENTS

- Assignment of customers to day of the week
- Time windows for pickup/delivery (soft, hard)
- Open and close times
- Load/unload (dwell) time

DATA REQUIREMENTS

- Geographic database, road networks
- Customer addresses and locations
- Travel times
- Vehicle location information
- Customer credit and billing information

(B) Constraints on Route Configuration

The route driven on any given day are subject to a number of constraints reflecting restrictions imposed by the vehicles, the driver schedules, customer's preferences or the geography of the service region. A partial list of such constraints are given below.

- Start and end locations for the route/driver
- Start times of routes
- Intermediate breaks
- Weight and volume restrictions on vehicle load
- Loading constraints and restrictions

- Driver territories or delivery regions
- Natural or legal region boundaries
- Importance of balanced routes
- Use of fixed routes and fixed stops on route
- Modification of routes based on new incoming orders
- Special rules for certain road segments
- One-way streets
- Avoiding U-turns or other safety rules.

(C) Objective Function

It is very important to decide the nature of objective function driving the model. The role of objective function is to control the quality of the generated solution. Quality of the solution means that to what extent we are able to balance the contrasting requirement of operators and customers. Some of them are

- Distance traveled by vehicle
- Number of vehicles used
- Excess ride time of customers
- Deviation from specified desired time
- Generation of balanced route

1.3 PURPOSE OF THIS THESIS

In ADART we need to develop a set of decentralized algorithms to accomplish the following tasks :

1. Schedule advance trip requests

2. Modify schedules based on new trip requests and cancellations.

3. Continuously improve the routes of vehicles.

In this model of ADART system, vehicles begin their day with some preassigned routes, that are subject to change as more customer requests come in. Each customer that has already requested service will be assigned to some vehicle's route. The routes of the vehicles will be changing dynamically as new customer requests (or cancellations) are received, new vehicle join the system and some existing vehicles go out of the service. Initial routes are valuable in that they will serve as a basis for reoptimization. These initial routes for the vehicles can be constructed by the centralized algorithm based on the complete information about all the customers that have requested in advance. Possibly this centralized algorithm could be run on a vehicle computer that has received customer request information in a decentralized manner.

In this thesis we have devised computerized algorithms for the scheduling of the advance trip requests in an efficient manner. We have used construction based heuristics which are demand responsive. Demand responsiveness means that in the case of a low demand (less customer requesting for ADART service) periods, the emphasis is placed on the quality

of the service, whereas in the case of heavy demand the emphasis is placed to conserve the vehicle resources.

We have used heuristic rather than exact approaches. In fact, exact algorithms to solve multivehicle advance request dial-a-ride problems do exist. However, they are quite slow and cannot solve realistically sized problems. Moreover, these initial routes will be dynamically changing to accommodate new requests and cancellations. So it is far more important to have a schedule that is robust to further changes than to have a schedule that has optimized initial solution. Accordingly, good heuristic algorithms to obtain initial routes of the vehicles is a sensible and practical alternative to exact algorithms.

We have also developed some randomization approaches in order to generate more initial feasible solutions of better quality. Finally, we have done the taxi simulation to evaluate the benefits obtained by the approaches devised in this thesis.

Chapter 2 of this thesis gives an overview of past work on various versions of the dial-a-ride problems. Chapter 3 describes the Autonomous Dial-A-Ride Transit System, introduces a construction based heuristic algorithms for scheduling advance trip requests and present computational results of these approaches. Chapter 4 provides modification based on randomization approaches and presents computational results of these algorithms.

CHAPTER 2 REVIEW OF ALGORITHMS FOR DIAL-A-RIDE PROBLEMS

2.1 INTRODUCTION

This chapter reviews earlier work that has investigated the dial-a-ride problem. The discussion consists of two parts. In Section 2.2, the immediate-request dial-a-ride problem is discussed. In Section 2.3, the advance-request version is discussed.

2.2 IMMEDIATE-REQUEST DIAL-A-RIDE PROBLEM

The immediate-request dial-a-ride problem involves dispatching a fleet of vehicles in response to customer demands that require immediate service. When dial-a-ride agency receives a new customer request, its vehicles are either in an idle state, i.e., waiting for further assignments, or are busy executing tasks that have been assigned to them previously. Similarly, customers in the system are either on board some vehicle on the way to their destination or are still waiting to be picked up. With the current status of vehicles and customers known, the problem is to determine which vehicle should serve the new customer and to develop a new route and schedule for the responsible vehicle.

Wilson and others [1, 2] have examined this problem in their pioneering work associated with the Rochester, New York Dial-A-Ride Demonstration Project. In [3], Psaraftis introduced an algorithm which can solve the single-vehicle immediate-request problem optimally using dynamic programming,

Psaraftis and Tharakan [4] later extended this approach to the multi-vehicle case and in [5] Psaraftis enhance the dynamic programming algorithm by including the capability of handling time window constraints. We now discuss these approaches in detail.

2.2.1 An Assignment Algorithm (Wilson et al. [1,2])

This algorithm selects the best way to incorporate the new customer into the existing vehicle route which is called "the provisional route". All possible combinations of insertions of the new customer into the provisional route are examined and preferences are determined by using a cost criterion which is a weighted sum of customer disutility and a vehicle resource function.

Customers are divided into classes depending upon the type of service demanded. Different classes of customers have different disutility functions. Let D_{ij} be the disutility for the ith customer in class j. Then D_{ij} is defined as :

$$D_{ij} = a_j W_i^2 + b_j R_i^2 + c_j P_i^2 \qquad (2\text{-}1)$$

where a_j, b_j and c_j are constants which reflect the particular service preferences of customers in class j.

W_i is the waiting time between the instant when a request was made and the actual pick-up time.

R_i is the time between a passenger's pick-up and delivery, i.e., the ride time.

P_i is the time between the promised pick-up time and the actual pick-up time.

The objective function, Z, that determines the relative merits of different insertions for a new customer p (in class q) is the sum of three parts :

1) The new customer p's own disutility : D_{pq}

2) The marginal disutility of other customers caused by the insertion of the new customer into the provisional route :

$$\sum_i \sum_j (D'_{ij} - D_{ij})$$

where D'_{ij} is the disutility for customer i in class j after the insertion of customer p into the route.

3) The cost in system resources :

$$(TL'_v - TL_v)\,(d.\overline{TL} + e.TL_v)$$

where d and e are parameter.

TL_v and TL'_v are the tour length for vehicle v before and after the insertion of customer p.

$\overline{TL}$ is the mean tour length for all active vehicle.

The system resource function is designed to conserve system resources, i.e., vehicle time, in relative response to the changing pattern of demands and to equalize workload among all vehicles.

Thus, the total objective function used in determining the desirability of an insertion is :

$$Z = D_{pq} + \sum_i \sum_j (D'_{ij} - D_{ij}) + (TL'_v - TL_v)\,(d.\overline{TL} + e.TL_v)$$

After Z values are evaluated for all possible insertions, the algorithm selects the one which has the minimal Z value as the best insertion. The new customer is then incorporated into the schedule according to the selected insertion.

Dynamic Programming (DP) Approach by Psaraftis [3]

Psaraftis has developed an algorithm which can solve the single-vehicle immediate-request problem with a linear objective function to optimality using dynamic programming. In [4], he further extended the approach to solve the multi-vehicle immediate-request problem. In this section, we will describe in detail the single-vehicle DP formulation and its extension to the multi-vehicle case, both within the immediate-request environment.

(i) Single-vehicle case

Assume that a dial-a-ride agency receives a customer request at time t = 0 and the vehicle is located at A at that time with none or some customers on board. The problem is to develop a new vehicle route which includes the new customer and minimizes a generalized objective function. The generalized objective function is expressed as follows :

$$\text{minimize } w_1 \cdot \sum_j T_j + w_2 \sum_i (\alpha.WT_i + (2-\alpha).RT_i) \qquad (2\text{-}2)$$

where T_j is the duration of the jth leg of the trip.

WT_i is the waiting time of customer i.

RT_i is the ride time of customer i.

w_1, w_2, α are weights associated with each term which can be specified by the user.

$0 \le \alpha \le 2$

The term $\sum_j T_j$, represents the time to serve all N customers (including the new one). The other term, $\sum_i (\alpha.WT_i + (2-\alpha).RT_i)$, represents the total

customer disutility which is a linear combination of the time each passenger waits for the vehicle and of the time he spends inside the vehicle until his delivery.

Constraints on the problem include : (a) vehicle capacity constraints (b) the maximum possible position shift from the customer's original standing in the call list. This second constraint acts as a "service guarantee" for each customer.

States in the DP formulation are represented by the vector $(L, K_1, \ldots, K_N)$ where

a) L : Point the vehicle is currently visiting. It is assumed that:

Between 1 and N : Vehicle is at origin of customer L

L = Between N+1 and 2N : Vehicle is at destination of customer L-N

2N+1 : Vehicle is at starting point A.

b) K_j : "status" of customer j (j = 1, ... , N). It is assumed that :

3 : Customer j has not been picked up so far

K_j = 2 : Customer j is in the vehicle

1 : Customer j has been delivered.

Stage variable n is zero when the vehicle is at the starting point , one at the first pick-up stop and so on, until n = 2N at the last delivery stop.

Based on the above formulation, the DP algorithm can solve the single-vehicle problem to optimality. Due to the exponential growth in computation time with respect to the number of customers involved, the algorithm can solve only small problems, i.e., 8-10 customers, within

reasonable computer time. For example, it takes 2.7 seconds of CPU time (VAX 11/782) for the case of N = 5, 46.8 seconds for N = 7 and 591.4 seconds for N = 9.

(ii) Multi-vehicle case

In the multi-vehicle problem, Psaraftis and Tharakan [4] have designed an assignment procedure which determines which vehicle should serve the new customer. After such an assignment is made, the untravelled portion of the chosen vehicle's route is reoptimized with the new customer included using the single-vehicle algorithm. The assignment procedure employs an objective function which determines the desirability of assigning the new customer to a vehicle. The objective function can take two forms :

(a) MINIMAX : The objective here is to minimize the latest time for all vehicles to complete their assigned route.

$$\text{minimize } \max \left[\sum_j T_j^k \right]$$

where T_j^k is as defined in (2-2) for the kth vehicle.

(b) MINISUM : The objective here is to minimize the sum, evaluated over all vehicles, of expressions similar to (2-1). It can be written as follows :

$$\text{minimize } \sum_k V_k$$

where V_k represents the expression in (2-2) for vehicle k.

The complete multi-vehicle algorithm can be outlined as follows :

Let Z denote the adopted form of the objective function.

Step 1 : For all vehicles j (j = 1, ... , m), compute V_j, the objective function for the jth vehicle. If Z = MINIMAX, set $Q = \max_j V_j$. This is an initialization step which does not involve the new customer.

Step 2 : Tentatively assign the new customer to each vehicle j and evaluate the prospective V_j'. After V_j' is obtained, do the following : If Z = MINIMAX and $V_j' \leq Q$, the new customer is assigned to vehicle j. The assignment procedure terminates. Otherwise, obtain V_j' for another vehicle.

Step 3 : If Z = MINISUM, then the new customer is assigned to that vehicle j for which $(V_j' - V_j)$ is minimized. If Z = MINIMAX, then the new customer is assigned to vehicle j for which max (V_j', Q) is minimized.

This assignment procedure is optimal if there is only one new customer and changing previously omitted assignments is not allowed. Like the single-vehicle algorithm, this multi-vehicle algorithm is limited by the exponential growth in computation time with respect to the number of points unvisited by each vehicle at the time of the new request. For large problems, the algorithm can be used under a constraint which limits the number of points to be considered for resequencing at the time of appearance of the new customer.

2.3 ALGORITHMS FOR THE ADVANCE-REQUEST DIAL-A-RIDE PROBLEM

In this section we will describe algorithms for the advance-request dial-a-ride problem for multi-vehicle case.

(a) NEIGUT/NBS algorithm [6]

The NEIGUT/NBS algorithm is aimed at maximizing vehicle productivity by determining the minimum number of vehicles necessary to provide dial-a-ride service. The algorithm uses the concept of a "base-trip" which is defined as a vehicle route starting from a customer's origin and ending at his destination. The base-trip customer serves as a seed in attracting other customers to share the ride with him. By packing as many customers as possible into a base trip, the algorithm forms a vehicle sub-tour (partial schedule) which starts with the first pick-up of a customer when the vehicle is empty and ends after delivering the last customer on board in the expanded base-trip. A vehicle's whole-day schedule is then constructed by linking different sub-tours together. Not all customers have to be served since the algorithm assumes that supplemental service, e.g., taxi, is available and might be less expensive than using dial-a-ride vehicles.

The algorithm also seeks to meet a specified quality of service for all customers. Customers are guaranteed to be served within given time intervals and their maximum on-board times cannot exceed a prespecified limit. No customer disutility or inconvenience is assumed in the model.

The algorithm starts by treating every customer as a base-trip and for each base-trip, it evaluates the possibility of insertion of another customer into the base-trip. For example, assume that for the base-trip customer k, p_k and d_k denote the pick-up and delivery of customer k. To insert another customer,

say, i to this base-trip, two possible permutations are examined : p_k, p_i, d_i, d_k, and p_k, p_i, d_k, d_i. Of all possible permutations (there might be more than two if the base-trip has been expanded already), the permutation selected is the one that is feasible in terms of service time constraints and maximum on-board time constraints and that minimizes the travel time from p_k to d_k. This procedure is repeated until no more customers can be added to the base-trip. The partial schedule formed is then assigned to a vehicle by linking it with other partial schedules already assigned to that vehicle.

The process of constructing partial schedules is repeated over the remaining set of unassigned customers until all customers are included in the vehicle schedules. It is to be expected that the routes generated first are the most productive and the routes generated last tend to be inferior in this respect. As a result, vehicle workloads tend to be unevenly distributed.

(b) Parallel insertion [7]

Roy et al. [7] develop an insertion-oriented algorithm for dial-a-ride system in Canada which provide transportation service to the handicapped. The algorithm guarantees a quality of service similar to that of the NEIGUT/NBS approach. Each customer is allowed to specify either a desired departure time or desired delivery time. Deviation from the desired service time is kept below a specified limit. The algorithm also places a upper bound on a customer's excess ride time. The customer disutility function is taken to be a linear weighted sum of service time deviation and excess ride time. Without considering customer disutility at first, the algorithm constructs an initial set of vehicle routes and schedules by sequentially inserting customers to the routes that have been built up from previous insertions. After all customers have been inserted into the initial set of routes, the algorithm then considers exchanging customers between vehicles so that customers' disutility can be reduced.

Before considering the insertions of customers, a concept of "neighbor" is introduced to group together neighboring customers who are likely to be served by one vehicle. For example, customer k is considered to be a neighbor of customer i if

(a) $LPT_k - EDT_i \geq d(-i, +k)$ and $d(-i, +k) \leq MAXSEP$

where MAXSEP is a pre-determined constant

or

(b) $LDT_k - EDT_i \geq d(-i, -k)$ and $d(-i, -k) \leq MAXSEP$

and $d(+i, +k) + d(+k, -i) < 1.4.d(+i, -i)$

The constant 1.4 is a parameter which approximately defines an ellipse around the stops. A "time-horizon" concept is also used to define the set of customers considered simultaneously a the candidates for the next insertion. The time-horizon is defined by a time window and a maximum number of customers allowed in the horizon.

The algorithm starts by sorting the customers in ascending order with respect to their desired service time. Initialization occurs by identifying those customers whose desired service time falls within the specified horizon. For each customer in the horizon, find his neighbors satisfying the relationship (a) or (b) above. Initialize one or several routes with groups of neighboring customers. Then, repeat the following procedure :

Step 1 Update the horizon, i.e. move the time window defining the horizon later in time and introduce more customers into the horizon.

Step 2 Attempt to find the best route for a chose customer d in the horizon. If it is infeasible to insert customer d, then relax the time

windows of customer d and try again. If still not feasible, initialize a new route for customer d.

Step 3 Attempt to insert the neighbors of customer d in the horizon into the same route to which customer d is inserted.

Step 4 If all customers are assigned to routes, go to Step 5. Otherwise, go to Step 1.

Step 5 Attempt to reduce customer disutility by exchanging customers between vehicles. Start with the customer who has the highest disutility and repeat the procedure until no improvement can be realized.

In [7], it is not clear that the algorithm will eventually provide a feasible solution in terms of satisfying service quality guarantees when time window constraints are relaxed in Step 2 above. In other words, the exchange procedure in Step 5 does not guarantee re-establishment of those relaxed constraints after exchanging customers between vehicles.

Test results of the algorithm are obtained for actual problems in Montreal and Sherbrooke in Canada. The results show that the algorithm can provide smaller cost schedules and better service quality when compared with the schedules produced by the dial-a-ride agencies. A case of 451 customer requests and 31 vehicles required 472.3 seconds of CPU time (type of computer used is not known).

CHAPTER 3 AN ALGORITHM FOR ADVANCE REQUEST SCHEDULING FOR ADART

3.1 INTRODUCTION

This chapter describes a heuristic algorithm, Advanced Dial-A-Ride With Time Windows (ADARTW), for the advance-request scheduling for ADART with service-quality constraints that include time windows. Service quality constraints refer to guarantees that (i) each customer's ride time will not exceed a pre-specified maximum, and (ii) the time of pick-up or delivery of a customer will not deviate from the most desired time by more than a pre-specified amount ("the time windows"). The back bone of our approach is the model proposed by J. J. Jaw [8].

In Section 3.2, we will describe about the ADART service. In the later sections, we will describe the structure of ADARTW and the heuristic techniques that it uses. These are to a large extent dictated by the operating scenario within which the algorithm was conceived. This scenario is described in Section 3.3, which also defines the mathematical notations used in the subsequent sections. Section 3.4 presents an overview of ADARTW. Section 3.5 describes the search for feasible insertions of customers into vehicle work-schedules. Section 3.6 describes the optimization procedure used to assign customers to vehicles and to fix pick-up and delivery times. Finally, Section 3.7 describes the computational results of the algorithms discussed in the earlier sections.

3.2 ADART FEATURES

From the customer's point of view, ADART costs more than a bus but provides a better service. It costs less than a taxi but provides a lesser service. It ignores the off-the-street occasional users. ADART has four salient features or advantages: subscription use, many-to-few service, convenience, and reliability.

From the operator's point of view, ADART's important features include cashless/checkless revenue collection, fully automated dispatching, on-board information, command and control systems, high driver productivity, low operating overhead, and easy adaptability to demand. Fully automated system assigns trips to vehicles, devises itineraries, and plans routes *without any human intervention;* and has its dispatching hardware and software *on board the vehicle.*

Each vehicle's computer communicates with its vehicle's driver and other computers. Having no central control, an ADART fleet behaves like a swarm of ants doing their work with no one in charge. This is not to suggest that an ADART operation lacks central management. Vehicle and data maintenance, technical trouble-shooting, service quality control, pricing, hiring, fixing, automatic call distribution, and accounting are a few examples of functions best served by ADART's central management. ADART has negligible centralized intervention (mechanical or human) between the time a customer requests service and the time he receives it.

The ADART service targets recurring trips, which account for quite high percent of urban travel. These include trips to attraction centers for work, shopping, and personal purposes. ADART targets these recurring trips because they form a lucrative market. They are serviceable with a many-to-

few operation and provide "repeat business," the life blood of an enterprise profiting on high volume.

3.2.1 Types of DART Service

Conventional DART service falls into one of the three categories: many-to-one, many-to-few, and many-to-many. All three imply *two-way* service.

Many-to-One (MTO)

The simplest service to provide, it transports passengers to or from many locations (e.g., residences) to one location (e.g., shopping complex). The user of MTO service has only to provide one trip-end location from the "many," the other end of his trip is always the "one."

Many-to-Few (MTF)

In contrast to MTO service, MTF serves more than one attraction center, and is harder to provide.

Every city has several market segments of its traveling public that MTF service can target. Since many trips go from home to some attraction center, MTF coverage is high enough to attract sufficient ridership, which maintain vehicle load factors that sustain service at moderate cost. This makes MTF the most cost-effective form of DART.

Many-to-Many (MTM)

In fact, MTM is a more general case of MTF. In this both trip ends are scattered.

3.2.2 Multi-Vehicle Service Areas

The most productive dial-a-ride systems allow more than one vehicle to pick-up passengers within the same service area. Attempting to simplify dispatching, the most primitive demand-responsive systems partition a service area into non-overlapping sub-areas, and assign sub-area coverage exclusively to a single vehicle.

- **Single Vehicle Coverage.** Given that service reliability is a feature that customers value most, it is dangerous to assign exclusive coverage of a geographic sub-area to one vehicle.

- **Multi-Vehicle Coverage.** By assigning multiple vehicles to the same service area, DART can assign a trip to the vehicle that most efficiently serves it. This assignment depends on the present and planned locations of all vehicles.

3.2.3 Fully Automated Dispatching

A fully automated dispatching (FAD) system is one that can field a customer's request for service, schedule a vehicle to provide that service, and optimally route the vehicle *without any human interaction*. That is, the customer is the only human involved in the entire process of requesting a ride, scheduling arrivals and routing the vehicle. There is no "telephone center" to receive calls nor any "centralized dispatch" to assign trips to vehicles. The driver merely obeys the driving directions he receives from his vehicle's computer.

The vehicle's on-board computer receives a customer request, inserts this request into the vehicle's schedule, and plans an optimal route to accomplish the schedule. Furthermore, if beneficial to do so, the computer may pass the request off to another DART vehicle, without the drivers

knowing it. The DART vehicle's computer controls the vehicle's route by continuously informing the driver of the next required change of direction.

3.2.4 Routing and Scheduling

Using every free computer cycle, the routing and scheduling system (RSS) continually works to improve the planned route of the vehicle to fulfill its outstanding trips. The RSS's goal is simply to devise an itinerary for the vehicle to pick-up and deliver would-be passengers according to their origin/destination and time-window requirements at minimum cost.

After composing an excellent route for its vehicle, the RSS might receive a new trip, whose inclusion destroys the viability of the planned route. Another complication is that the computer has limited time to find a feasible solution.

Trip-Vehicle Assignment

Whenever a user calls the ADART phone number, the vehicle's computer receives the call, prompts trip data from the customer. This vehicle's computer then initiates an "auction" for the trip.

This auction begins with the initiating vehicle/computer (VC) estimating the "marginal cost" (e.g., expected miles/trip) of including that trip into its vehicle's schedule. The computer figures this cost by inserting the trip into its vehicle's planned route in a way that assures the vehicle meeting all its scheduled arrivals.

The initial VC then broadcasts this cost and the trip's requirements to every vehicle serving the same sub-area. Each of these latter VCs makes its own estimate of including the trip in its own itinerary. Any VC whose estimate is lower than the initial VC's responds with its own cost. The others remain silent.

The initial VC then informs the VC with the lowest cost of its responsibility for the trip. This responsible VC enters the trip into its scheduling system, which grinds away toward an optimal route to serve all its outstanding trips. All other VCs ignore the trip.

In technical parlance, the vehicles' computers collectively assign the new trip to a "cluster" belonging to the responsible vehicle, thus leaving each vehicle's computer to solve only one small traveling salesman problem (TSP). Furthermore, all the vehicles' computers work on their particular TSP in parallel. This decomposes an otherwise huge, daunting problem into several easier small problems - all solved simultaneously - and enables an ADART operation to keep up with even the largest demand surges.

Note that no human participates in trip - vehicle assignment, routing or scheduling - not the driver, not the customer, and certainly not any dispatcher.

Trip-Vehicle Reassignment

A further important application of this auction algorithm occurs when a vehicle's RSS notes that its vehicle's projected productivity has dropped below some critical threshold. Here, a "problem trip" is uncovered by excluding each trip from the schedule and learning the maximum cost saving. The trip that saves the most is put up for auction, so that a better positioned vehicle might service it. This simple on-the-fly expedient improves productivity and reduces service delays.

Vehicle Failure

The same algorithm supports fail-soft measures when a vehicle unpredictably goes out of service. Here, all the disabled vehicle's trips are auctioned off to the remaining vehicles.

3.3 OPERATING SCENARIO

We will first present the definitions and notations used in this chapter. The various inputs to the system are :

DPT_i (DDT_i) : the desired pick-up (delivery) time of customer i

DRT_i : the time it would take a vehicle to go directly from the origin to the destination of customer i, [$DRT_i = D(+i, -i)$]

D(x, y) : the time it takes a vehicle to go from point x to point y (using fastest route)

WS_i : the maximum acceptable deviation of customer i from his desired pick-up or delivery time ($DV_i \leq WS_i$)

Variables to the system are defined as follows :

N : the number of customers on the subscriber list

m : the number of available vehicles

EPT_i (EDT_i) : the earliest possible time at which the pick-up (delivery) of customer i can be made

LPT_i (LDT_i) : the latest possible time at which the pick-up (delivery) of customer i can be made

APT_i (ADT_i) : the time when the pick-up (delivery) of customer i will *actually* take place according to the schedule

ART_i : The actual ride-time of customer i, $ART_i = ADT_i - APT_i$.

+i (-i) : the event "pick-up (deliver) customer i", the indication "+i" ("-i") is also used to denote the *point of origin* (*destination*) of customer i

DV_i : the deviation in the time schedule of customer i from his desired pick-up or delivery time [for DPT-specified customers $DV_i = APT_i - DPT_i$; for DDT-specified customers $DV_i = DDT_i - APT_i$]

d: the number of stops (pick-ups and deliveries) in a schedule block.

$SLACK_j$: the duration of vehicle slack time before schedule block j. If there are n schedule blocks, $SLACK_{n+1} = \infty$.

The dial-a-ride system with which ADARTW is designed to operate is assumed to have a number of characteristics. The most important among those is that each of the system's customers is asked and willing to specify either a desired pick-up time (DPT) at his origin or a desired delivery time (DDT) at his destination, but not both. This implies that the customer is able to decide for himself whether he is constrained by a pick-up time or by a delivery time in his intended trip. For example, most individuals are constrained in the morning by a desired "delivery" time (e.g., the time one has to arrive at the work place) and adjust their "pick-up" time (e.g., the time when they leave their homes) accordingly. In a similar fashion, a dial-a-ride system's customer who specifies a desire to be delivered at a commuter train station (or an outpatient clinic) by time X, will be a "DDT-specified" customer. In our system, this customer would rely on the operator of the system to tell his at what time he will be picked up from his origin so that he can be delivered at his destination by time X. The reverse is, of course, true for DPT-specified customers (e.g., "I can be picked up from the shopping mall at 2 P.M. for transportation to my house"). Normally, one would expect a preponderance of DDT-specified customers in the morning and DPT-specified customers in the afternoon and evening.

A second and related assumption is that DPT-specified customers will be asked to give as their DPT, the earliest time at which they can be picked up. Similarly, DDT-specified customers will give the latest time at which they can be acceptably delivered at their destination as their DDT. This actually implies no loss of generality, but is particularly convenient for the algorithm, since the time-window during which a DPT- (DDT-) specified customer can be picked up (delivered) can be defined as beginning (ending) with the specified DPT (DDT).

In dial-a-ride system in question one would certainly expect a commitment of quality of service on the part of the system's operator. Consider, for instance, a DDT-specified customer who lives 15-minutes away (by car) from a community center and who desires to be at that center by 10 A.M. It would clearly be unreasonable to deliver that customer at the center at, say 8 A.M. or to offer him a 90-minutes circuitous ride - picking up and/or delivering many other customers on the way. For these reasons it will be assumed that the system will operate under three types of service quality constraints :

(i) No DPT- (DDT-) specified customer will be picked up (delivered) earlier (later) than his DPT (DDT).

(ii) No customer's actual ride time will exceed a given maximum ride time for that customer - the maximum ride time being specified as a function of the direct origin-to-destination ride time for that customer.

(iii) The difference ("time deviation") between the actual pick-up (delivery) time and desired pick-up (delivery) time of a customer will not exceed a given maximum for DPT- (DDT-) specified customers.

The values of the maximum ride-time and of the maximum time-deviation can either be determined unilaterally by the system's operator and applied universally or, can be left open to negotiation between the operator and each

individual customer. In the former case, an operator might advertise, for example, that a customer's ride time would "under no circumstances" exceed twice his direct ride time and that he would be delivered (picked-up) no earlier (later) than 20 minutes prior to (after) his desired delivery (pick-up) time.

3.3.1 The problem

The version of DARP solved by ADARTW can now be summarized as follows : Given a subscription list of N customers, each specifying either a DPT_i or DDT_i (i = 1, 2, ..., N) and a fleet of m vehicles, find an effective allocation of customers among vehicles and an associated time schedule of pick-ups and deliveries such that :

1, For all customers i :

$$ADT_i - APT_i \leq MRT_i \qquad (3.1)$$

2. For DPT-specified customers :

$$DPT_i \leq APT_i \leq DPT_i + WS_i \qquad (3.2)$$

3. For DDT-specified customers :

$$DDT_i - WS_i \leq ADT_i \leq DDT_i \qquad (3.3)$$

Several comments are in order concerning this formulation :

(a) We have not yet specified what is our measure of effectiveness (see Section 3.4 and 3.6).

(b) It may prove *infeasible* to serve some of the N customers with the given vehicle resources and service-quality constraints.

(c) MRT_i, the maximum ride time for customer i, will normally be specified as a function of the direct ride time, DRT_i. In our work we have used :

$$MRT_i = A + B \cdot DRT_i \qquad (3.4)$$

where A and B are user-specified constraints (e.g., A = 10 minutes, B = 1.5). A reasonable alternative might be :

$$MRT_i = \begin{cases} DRT_i + A & \text{if } DRT_i \leq T_0 \\ B \cdot DRT_i & \text{if } DRT_i > T_0 \end{cases}$$

where, again A, B and T_0 are operator-specified constraints (e.g., A = 10 minutes, T_0 = 20 minutes, B = 1.5) satisfying the relationship $T_0 + A = B \cdot T_0$. Other functional forms can, of course, be used to specify MRT_i, if desired.

In conclusion, ADARTW is designed for use under the following scenario : The system's operator would advertise the dial-a-ride service and the service quality guarantees that will be offered. Customer will call and specify origin, destination and a desired DPT (= EPT) or DDT (= LDT). Just before the start of the service, initial routes will be generated by the computer on board the vehicle. These routes can be generated using ADARTW algorithm based on complete information about all the customers that have requested service in advance. Routes of all the vehicles will be changing dynamically as new customer requests (or cancellations) are received, new vehicles join the system and some existing vehicles going out of the service. Initial routes are valuable in that they will serve as a basis for re optimization.

Before proceeding to the description of the algorithm a number of additional assumptions in our scenario will now be listed.

a) The capacity of vehicles is assumed finite and it is not necessarily the same for all vehicles.

b) Dwell times - the amount of time needed to pick up and deliver customers - can be non-zero and different customers are allowed to have different dwell times. It should be noted that non-zero dwell times can be handled by adding them to the distance matrix in a way consistent with the definitions of APT and ADT. We have taken dwell time to be 5 minutes for all customers by adding it to the time matrix.

c) A vehicle is not allowed to wait idly when it is carrying passengers. For example, it is not permissible in ADARTW to have a vehicle, with one or more passengers on board, arrive at the pick-up point of a DPT-specified customer i prior to DPT_i and then wait idly until time DPT_i to pick up i (remember that due to (3.2) customer i cannot be picked up earlier than DPT_i). Such idle waiting periods by non-empty vehicles are accepted by some vehicle-routing algorithms with time window constraints. It is felt, however, that, in particular, such idle waiting would not be tolerated by dial-a-ride customers and ADARTW has been designed accordingly. Actually this can be viewed as a fourth service-quality constraint, imposed in addition to (i) - (iii) above.

d) Each vehicle can have different origin and different destinations. Also each vehicle may be available in different time slots in a day.

Finally, it is useful to define *availability* periods, *active* periods and *slack* periods for vehicles. As show in Figure 3.1 for a particular vehicle j, a vehicle can be unavailable during periods of a day (usually due to driver constraints, labor union agreements or vehicle maintenance requirements). An available vehicle can be either in a slack period (i.e., waiting idly) or it can be active (on the way to pick up it's first customer during an active period, transporting, picking up or delivering customers or returning to a depot). Note that because of assumption c) above, a vehicle cannot be in a slack period as long as it has even one customer on board.

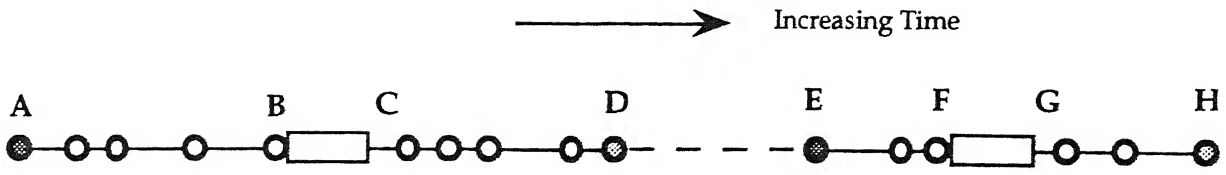


Figure 3.1

A typical schedule of a vehicle during service period AH.

Denotes Origin of a Vehicle

Denotes Destination of a Vehicle

Denotes Pickup or Delivery of a Customer

Denotes Slack Time of the Vehicle during interval BC and FG

Denotes Unavailability of the Vehicle during interval DE

3.4 Overview of the Algorithm

ADARTW is a heuristic algorithm that processes ride requests sequentially, inserting one customer at a time into the work-schedule of some vehicle until all ride requests have been processed. This section describes in qualitative terms how this procedure works.

Central to the process of assigning customers to vehicles are : a search for *feasible* insertions of customers into work-schedules; and a sequence of *optimization* steps designed to find the most desirable one among all the feasible insertions on each occasion. An insertion of a particular customer i into the work-schedule of a specific vehicle j is feasible only if it does not lead to violation of any service-quality constraints for customer i and for all other customers *already* assigned to vehicle j. The optimization steps deal with minimizing the additional "cost" due to inserting customer i into a vehicle's work-schedule. The cost-function that we use is a weighted sum of disutility to the system's customers (due to excess ride times and to deviations from the

most desirable pick-up or delivery times) and of system costs as represented by a function that quantifies the "consumption" of available vehicle resources.

Consider now a case in which there are N (advance-request) customer demands for service and m available dial-a-ride vehicles. ADARTW begins by indexing customers in the order of their "earliest pick-up times", EPT_i (i = 1, 2, ... , N), i.e., according to the earliest time at which they are expected to be available for a pick-up. The first customer in the sequence is the one with the smallest EPT_i (Section 3.5.1 shows how the EPT_i are computed).

The algorithm then processes the first, second, third, etc. customers in the list, one at a time (see also below), and assigns each customer to a vehicle until the last customer is exhausted. The processing of customer i goes as follows :

Step 1 : For each vehicle j (j = 1, 2, ... , m), in which a customer is already assigned.

(i) Find all the possible ways in which customer i can be inserted into the work-schedule of j (details in Section 3.5).

(ii) Find the insertion of customer i into the work-schedule of vehicle j that results in minimum additional cost (details in Section 3.6). Call this additional cost, $COST_j$.

Step 2 : If it is infeasible to insert i into the work-schedule of any vehicle, then add new vehicle in the system and assign the customer to it; otherwise assign i to a vehicle j^* for which $COST_{j^*} \leq COST_j$ for all j = 1, 2, ... , m.

The above is only the "generic" version of the algorithm. We have, in fact, developed a number of options which are available at various points in the procedure :

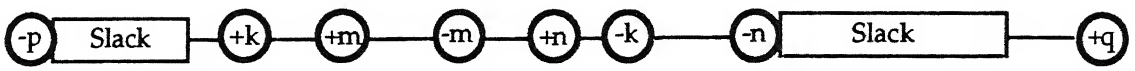
(a) Customers can be indexed and processed according to criteria other than EPT. For example, one can also process customers "backward" by ordering them according to their latest delivery time, LDT - with the customer having the largest LDT being processed first. Such alternative processing orderings can be used to generate several alternative solutions to any given problem instance.

(b) Instead of processing one customer at a time, the user can specify how many (yet unassigned) customers should be considered in Step 1 above. For example, if the number "5" is specified, the top 5 (in terms of their ordering index) unassigned customers will be considered on each occasion as candidates to be assigned next to a vehicle. It should be emphasized that each of the candidates is considered separately in Step 1, so that in Step 2 the one among (the 5, in our example) candidates with the smallest $COST_{j^*}$ will be assigned to vehicle j^*. We can term such a group of candidates as the candidate "pool" from which the next customer for insertion is selected. There exist two possible strategies for bringing unassigned customers into the pool. One strategy would be to bring a new customer into the pool every time a customer in the pool is inserted into some vehicle's work-schedule. In this way, a pool always maintains the same number of candidate customers in it. A customer will leave the pool either when he is selected for next insertion in Step 2 or when it is infeasible for any vehicle to carry him. Such a strategy is termed *immediate-refill.* An alternative is to let the candidate pool become smaller and smaller as customers in the pool are inserted, one at a time, to vehicle work-schedules. When the pool is finally exhausted, refill it with the next batch of customers in the list. This strategy is termed *periodic-refill.* Computational experience with both strategies will be reported later.

This multi-candidate option is provided for the purpose of improving the performance of the algorithm by making it less "myopic", i.e., by giving it an opportunity to select among more than one customer for the next assignment. The penalty, of course, is that as the number of candidate that are considered each time increases the efficiency of the algorithm decreases.

3.5 SEARCH FOR FEASIBLE INSERTIONS

We now turn to an outline of the steps taken to identify feasible insertions of customers into vehicle work-schedules. A notion which plays an important role in this respect is that of a "schedule block". This can be illustrated through Figure 3.2, which shows part of the work-schedule of a vehicle. A schedule block is a period of continuous active vehicle time between two successive periods of vehicle slack time. A schedule block always begins with an empty vehicle starting (either from a depot or after an inactive period) on its way to pick up a customer and ends with a period of slack time or at the end of the vehicle's entire work-schedule. In the case of a schedule block, a vehicle might become empty several times before the schedule block ends. Associated with a schedule block is a "schedule sequence" indicating the sequence of stops in the block and a "time schedule" indicating the time when each stop is scheduled to take place. For example, in Figure 3.2, the schedule sequence associated with the middle schedule block is (+k, +m, -m, +m, -k, -n) while the time schedule is (APT_k, APT_m, ADT_m, APT_n, ADT_k, ADT_n).



Schedule Block | Schedule Block | Schedule Block

Figure 3.2

Suppose now that we wish to examine whether a yet-unassigned customer i can be inserted into the work-schedule of vehicle j. The objectives of the search for feasible insertions are :

(i) To identify new feasible schedule sequences.

(ii) For each of the feasible schedule sequences to obtain a feasible time schedule.

ADARTW accomplishes these objectives by examining systematically and efficiently all possible schedule sequence associated with each and every schedule block on the work-schedule of vehicle j. For example, with respect to the middle schedule block of Figure 3.2, the possible schedule sequence involving the insertion of customer i are (+i, -i, +k, +m, -m, +n, -k, -n), (+i, +k, -i, +m, ... , -m), ... , (+k, +m, ... , -n, +i, -i). In all if there are d stops already on a schedule block, there are (d+2)(d+1)/2 possible schedule sequences that involve the insertion of a new customer into that schedule block, while adhering to the constraint that the "+i" stop must precede the stop "-i". In view of the maximum ride time and maximum time-deviation constraints of our version of DARP (relation (3.1)-(3.3)) only some (and possibly none) of the above possible sequences may be feasible.

More schedule sequences are possible if we consider the insertion of stop "+i" and stop "-i" into different - not necessarily consecutive - schedule blocks. Such insertions will result in merging all the schedule blocks between (and including) the ones where the pick-up and delivery are inserted. ADARTW considers both types of insertions, but uses different methods (see below) to test the feasibility of schedule sequences formed.

It should be noted that, in addition to inserting customer i into one of the already existing schedule blocks of any vehicle j, ADARTW will, naturally, consider creating an entirely new schedule block for vehicle j, as shown in

Figure 3.3, in order to accommodate customer i. For example, the first customer ever assigned to a vehicle will obviously always create a new schedule block. Such new schedule blocks are added to the list of existing schedule blocks to which ADARTW attempts to add new customers through the insertion procedure.

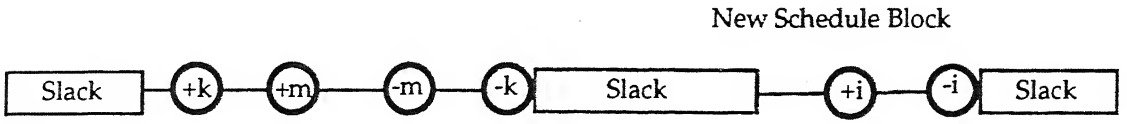


Figure 3.3

3.5.1 A Fast Screening Test For Insertions Within The Same Schedule Block

To facilitate the feasibility search, ADARTW includes a procedure that increases greatly its efficiency and is fundamental to its viability in dealing with large-scale problems. This procedure involves defining two time windows for each customer as follows :

For DPT-specified customer, we define :

$$EPT_i = DPT_i \tag{3.5}$$

$$LPT_i = EPT_i + WS_i \tag{3.6}$$

$$EDT_i = EPT_i + DRT_i \tag{3.7}$$

$$LDT_i = LPT_i + MRT_i \tag{3.8}$$

Note that (3.5) and (3.6) contain the same information as (3.2).

Similarly, for each DDT-specified customer, we define :

$$LDT_i = DDT_i \tag{3.9}$$

$$EDT_i = DDT_i - WS_i \quad (3.10)$$

$$LPT_i = LDT_i - DRT_i \quad (3.11)$$

$$EPT_i = EDT_i - MRT_i \quad (3.12)$$

(3.9) and (3.10) contains the same information as (3.3))

For any customer i, whether DPT- or DDT-specified, a set of necessary, but not sufficient conditions for feasibility is then provided by (3.13) and (3.14) :

$$EPT_i \leq APT_i \leq LPT_i \quad (3.13)$$

$$EDT_i \leq ADT_i \leq LDT_i \quad (3.14)$$

It should be noted that LPT_i need not be smaller than EDT_i.

The conditions (3.13) and (3.14) are particularly convenient to work with. The quantities EPT_i, LPT_i, EDT_i and LDT_i computed for all customers i = 1, 2, ... , N define "fixes" on the time axis within which the customer must be picked up and delivered. To understand the usefulness of these fixes let us return to the problem of checking the feasibility of inserting customer i into a particular schedule block "p" of vehicle j.

Let us consider such a schedule block and let us index the successive stops on the schedule sequence with the subscript r = 1, 2, ... , d. Note that from (3.13) and (3.14) we have an upper and lower bound for each element in the time schedule associated with our schedule block ((3.13) providing the bounds if the entry is an APT and (3.14) if the entry is an ADT).

For convenience let us now drop the indication of whether a particular stop on the schedule block is a pick-up or a delivery and use ET_r, AT_r and LT_r to denote earliest, actual (scheduled) and latest time, respectively, for stop r. For instance, in Figure 3.2, we would indicate EPT_k, APT_k, LPT_i by ET_1, AT_1 and

LT_1, respectively, and EDT_m, ADT_m, LDT_m by ET_3, AT_3 and LT_3, respectively, for the middle schedule block.

In ARDTW, we compute and store four statistics for each stop r (r = 1, ... , d) on each schedule block, defined as follows :

$$BUP_r = \min \left[\min_{1 \le t \le r} (AT_t - ET_t), SLACK_p \right] \quad (3.15)$$

$$BDOWN_r = \min_{1 \le t \le r} (LT_t - AT_t) \quad (3.16)$$

$$AUP_r = \min_{r \le t \le d} (AT_t - ET_t) \quad (3.17)$$

$$ADOWN_r = \min \left[\min_{r \le t \le d} (LT_t - AT_t), SLACK_{p+1} \right] \quad (3.18)$$

where $SLACK_p$ and $SLACK_{p+1}$ denote, respectively, the duration of the slack period immediately preceding and immediately following the schedule block p in question.

There is a very real intuitive meaning associated with the four quantities defined by (3.15)-(3.18). Specifically, BUP_r ($BDOWN_r$) represents the maximum amount of time by which every stop preceding but not including stop r+1 can be advanced (delayed) without violating the time-window constraints. Similarly AUP_r ($ADOWN_r$) represents the maximum amount of time by which every stop following but not including stop r-1 can be advanced (delayed). Essentially, the quantities BUP, BDOWN, AUP and ADOWN indicate by how much, at most, each segment of a schedule block (e.g., the segment that precedes the pick-up of the inserted customer i) can be displaced in order to accommodate an additional customer i.

The uses of the four statistics defined at each stop for checking the feasibility of an insertion differ depending on where in the schedule block the pick-up and delivery of the new customer are inserted. ADARTW divides all

insertions into four basic cases , which account for the total of (d+2)(d+1)/2 combinations mentioned earlier :

(i) Both the pick-up and delivery of customer i are inserted at the end of the *last* schedule block, i.e., they become the last two stops in the vehicle's work-schedule. We note that this is the only case where a new schedule block can be created.

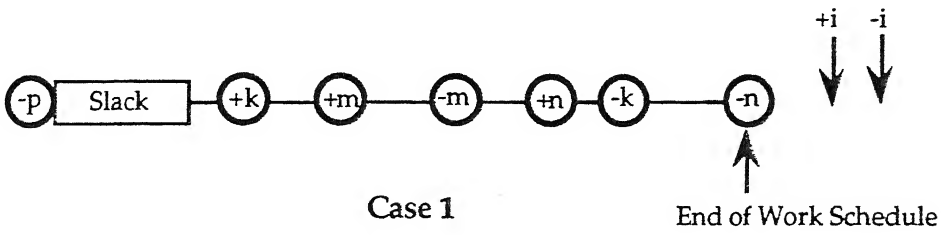
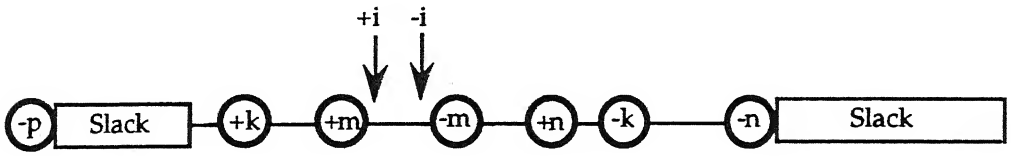


Figure 3.4

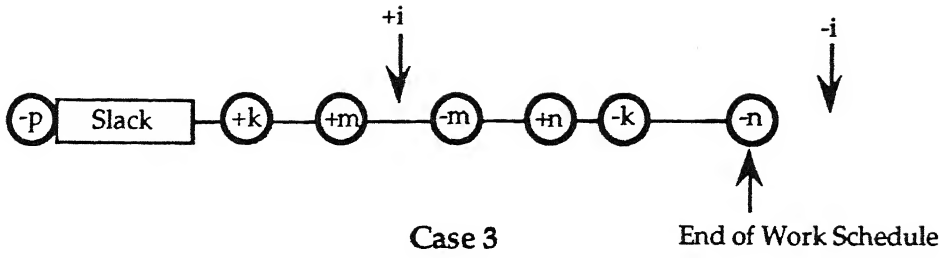
(ii) Both the pick-up and delivery of customer i are inserted between two consecutive stops on the schedule block. This includes the insertions of pick-up and delivery immediately before or after a slack period.



Case 2

Figure 3.5

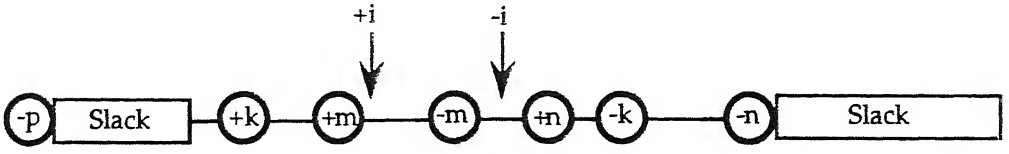
(iii) The pick-up of customer i takes place somewhere within the *last* schedule block while his delivery is inserted at the *end* of the vehicle's work schedule.



Case 3

Figure 3.6

(iv) The pick-up and delivery of customer i are separated by at least one other stop and the delivery of i is not the last stop in the vehicle's work-schedule.



Case 4

Figure 3.7

The details of the algorithm's logic for each one of these four cases are provided in Figures 3.8 to 3.11. Each flowchart depicts how it is determined whether it is feasible, as far as the time-window constraints are concerned, to insert customer i within a schedule block in the way defied by one of the four cases above.

Besides checking for violations of the time-window constraints, we have to check that no maximum-ride-time constraints are violated for the newly inserted customer and for the customers already in the schedule block. This can be done very easily by scanning through the list of these customers and comparing the respective actual ride-time and maximum available rime-time. Finally, vehicle loads at each stop between the inserted pick-up and delivery of customer i are checked so that vehicle capacity is not exceeded.

The notations used in the flow charts are given below :

p, q, r, s : Indices denoting visit sequence in a vehicle schedule block having d stops. They are used to define where the insertion of a new pick-up (+i) and delivery (-i) is made. Four cases of insertions can be defined as follows:

Case 1) 1 ..... p (= d) +i -i

Case 2) 1 ..... p +i -i q ..... d

Case 3) 1 ..... p +i q ..... d -i

Case 4) 1 ..... p +i q ..... r -i s ..... d

$T_i(T_{-i})$: The pick-up time (delivery time) of the new customer i.

ST_j : Scheduled visit time of stop j in the schedule block before customer i is inserted.

w_i : Amount of change in vehicle waiting time due to the insertion of customer i.

SHIFT : The amount of time by which a given time or a time schedule is being shifted.

ΔP_i (ΔD_i) : The extra vehicle time required to pick up (deliver) customer i.

PS(DS) : Shift in time schedule for all stops preceding stop +i (succeeding stop -i) after customer i is inserted.

MS : Shift in time schedule for all stops between +i and -i after customer i is inserted.

GT,ET,LT: Variables

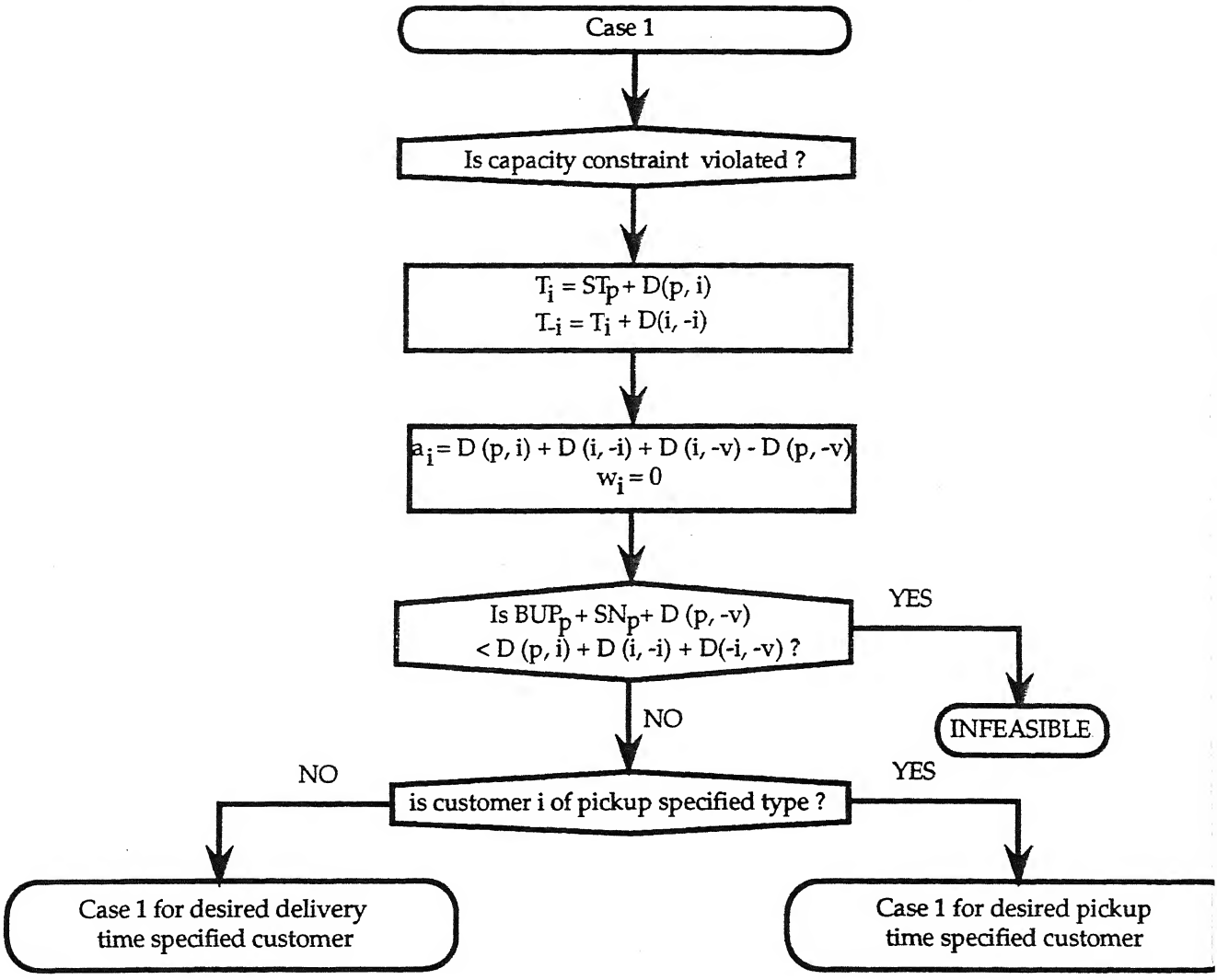

Case 1
Is capacity constraint violated ?
$T_i = ST_p + D(p, i)$
$T_{-i} = T_i + D(i, -i)$
$a_i = D(p, i) + D(i, -i) + D(i, -v) - D(p, -v)$
$w_i = 0$
Is $BUP_p + SN_p + D(p, -v)$ $< D(p, i) + D(i, -i) + D(-i, -v)$?
YES
INFEASIBLE
NO
NO
is customer i of pickup specified type ?
YES
Case 1 for desired delivery time specified customer
Case 1 for desired pickup time specified customer

Figure 3.8 (a)

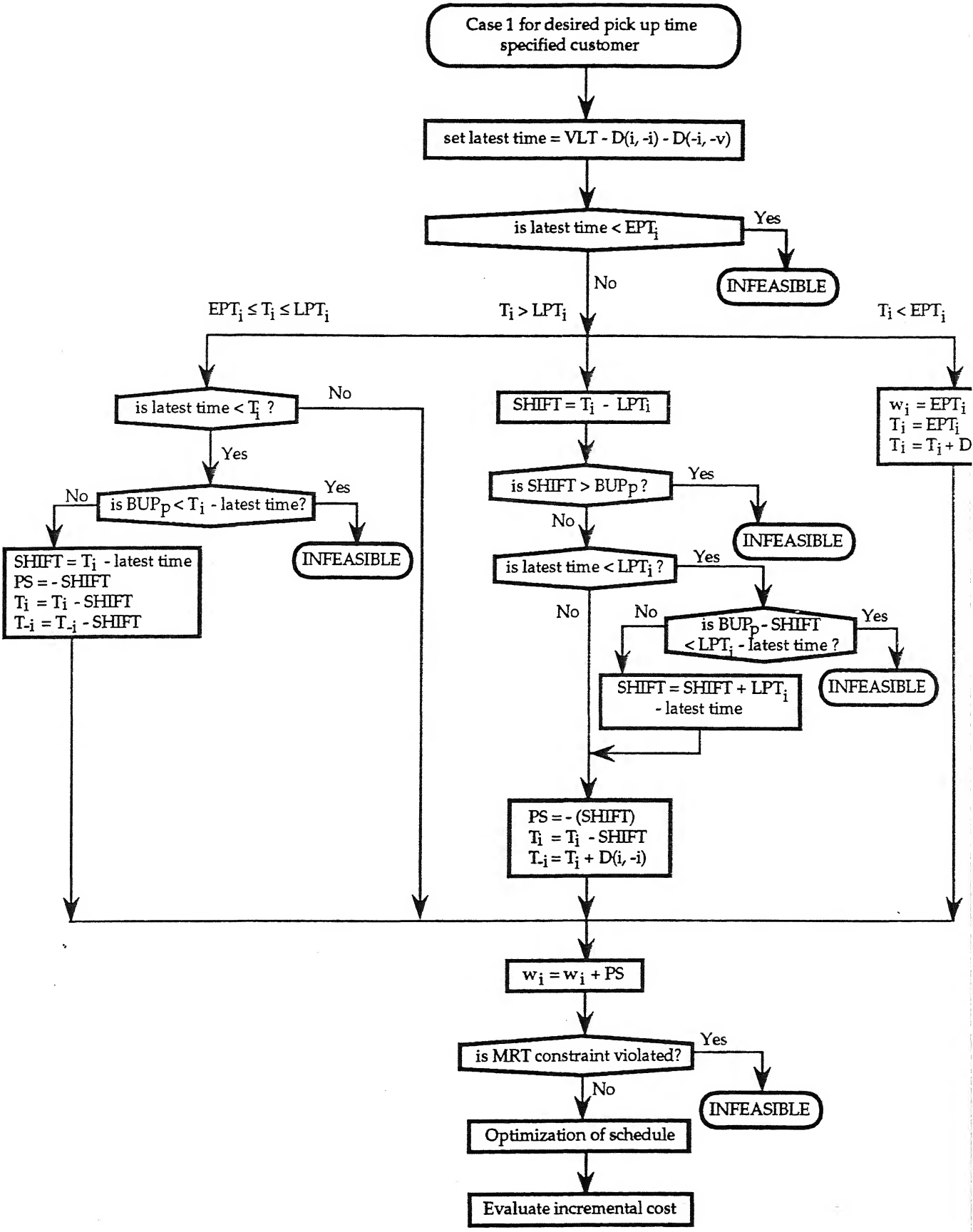

Case 1 for desired pick up time specified customer
set latest time = VLT - D(i, -i) - D(-i, -v)
is latest time < EPT_i
Yes
INFEASIBLE
No
EPT_i ≤ T_i ≤ LPT_i
T_i > LPT_i
T_i < EPT_i
is latest time < T_i ?
No
Yes
No
is BUP_P < T_i - latest time?
Yes
INFEASIBLE
SHIFT = T_i - latest time
PS = - SHIFT
T_i = T_i - SHIFT
T_-i = T_-i - SHIFT
SHIFT = T_i - LPT_i
is SHIFT > BUP_P ?
Yes
INFEASIBLE
No
is latest time < LPT_i ?
Yes
No
No
is BUP_P - SHIFT < LPT_i - latest time ?
Yes
INFEASIBLE
SHIFT = SHIFT + LPT_i - latest time
w_i = EPT_i
T_i = EPT_i
T_i = T_i + D
PS = - (SHIFT)
T_i = T_i - SHIFT
T_-i = T_i + D(i, -i)
w_i = w_i + PS
is MRT constraint violated?
Yes
No
INFEASIBLE
Optimization of schedule
Evaluate incremental cost

Figure 3.8 (b)

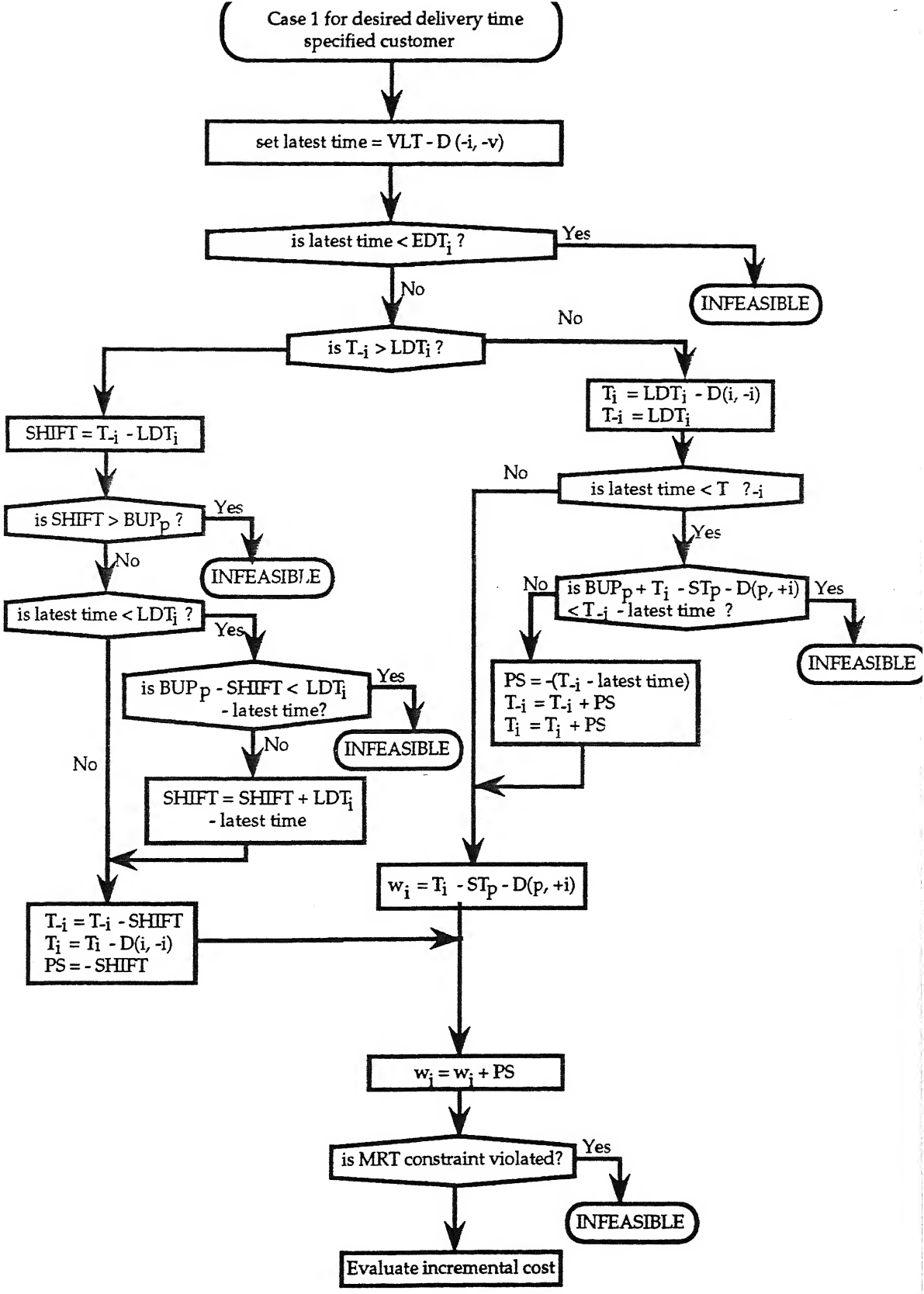

Case 1 for desired delivery time specified customer
set latest time = VLT - D (-i, -v)
is latest time < EDT$_i$?
Yes
INFEASIBLE
No
is T$_{-i}$ > LDT$_i$?
No
T$_i$ = LDT$_i$ - D(i, -i)
T$_{-i}$ = LDT$_i$
SHIFT = T$_{-i}$ - LDT$_i$
No
is latest time < T ?$_{-i}$
is SHIFT > BUP$_p$?
Yes
INFEASIBLE
No
Yes
No
is BUP$_p$ + T$_i$ - ST$_p$ - D(p, +i) < T$_{-i}$ - latest time ?
Yes
INFEASIBLE
is latest time < LDT$_i$?
Yes
is BUP$_p$ - SHIFT < LDT$_i$ - latest time?
Yes
INFEASIBLE
PS = -(T$_{-i}$ - latest time)
T$_{-i}$ = T$_{-i}$ + PS
T$_i$ = T$_i$ + PS
No
No
SHIFT = SHIFT + LDT$_i$ - latest time
w$_i$ = T$_i$ - ST$_p$ - D(p, +i)
T$_{-i}$ = T$_{-i}$ - SHIFT
T$_i$ = T$_i$ - D(i, -i)
PS = - SHIFT
w$_i$ = w$_i$ + PS
is MRT constraint violated?
Yes
INFEASIBLE
Evaluate incremental cost

Figure 3.8 (c)

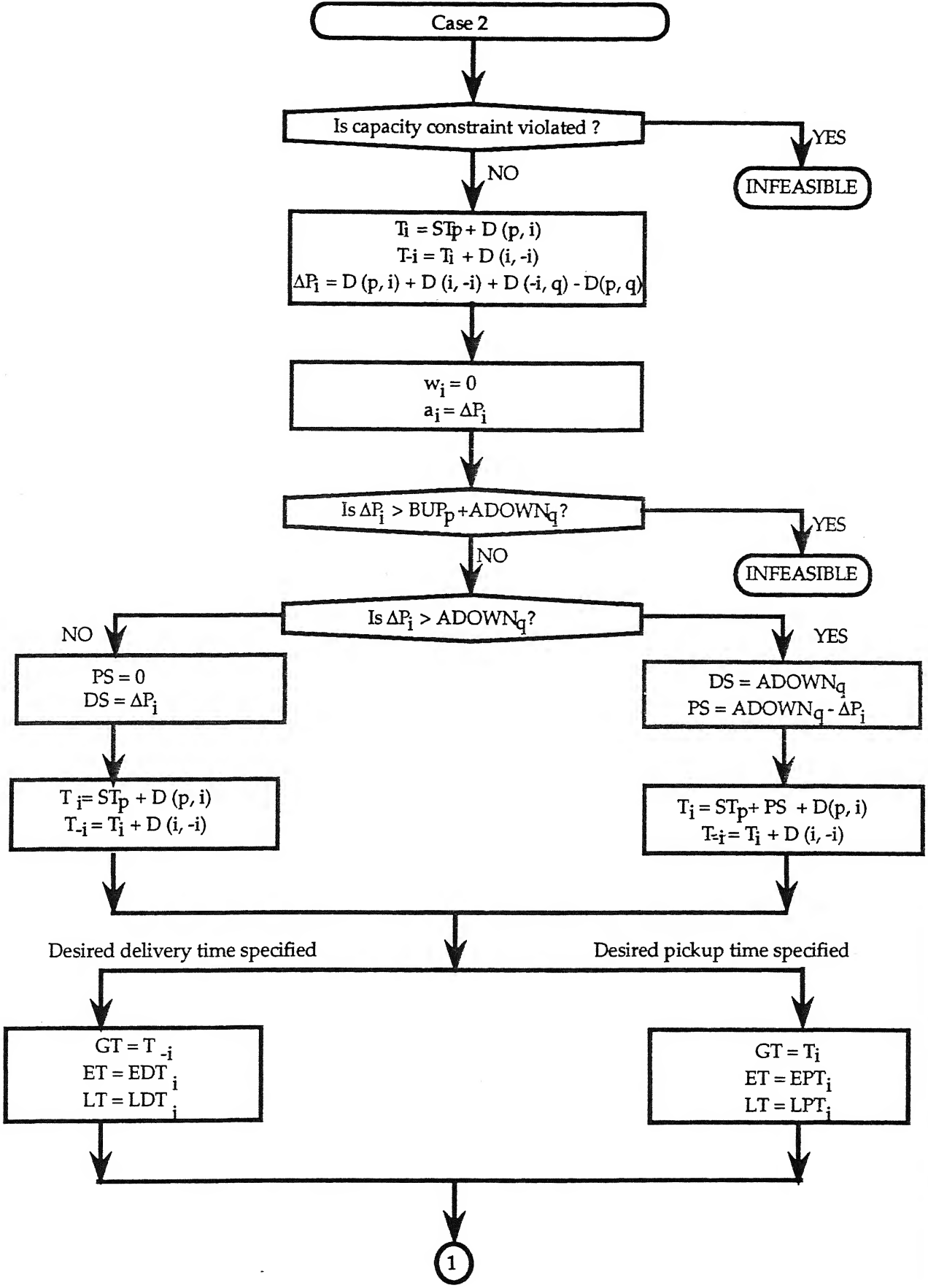

Case 2
Is capacity constraint violated ?
YES
INFEASIBLE
NO
$T_i = ST_p + D(p, i)$
$T_{-i} = T_i + D(i, -i)$
$\Delta P_i = D(p, i) + D(i, -i) + D(-i, q) - D(p, q)$
$w_i = 0$
$a_i = \Delta P_i$
Is $\Delta P_i > BUP_p + ADOWN_q$?
YES
INFEASIBLE
NO
Is $\Delta P_i > ADOWN_q$?
NO
YES
$PS = 0$
$DS = \Delta P_i$
$DS = ADOWN_q$
$PS = ADOWN_q - \Delta P_i$
$T_i = ST_p + D(p, i)$
$T_{-i} = T_i + D(i, -i)$
$T_i = ST_p + PS + D(p, i)$
$T_{-i} = T_i + D(i, -i)$
Desired delivery time specified
Desired pickup time specified
$GT = T_{-i}$
$ET = EDT_i$
$LT = LDT_i$
$GT = T_i$
$ET = EPT_i$
$LT = LPT_i$
1

Figure 3.9 (a)

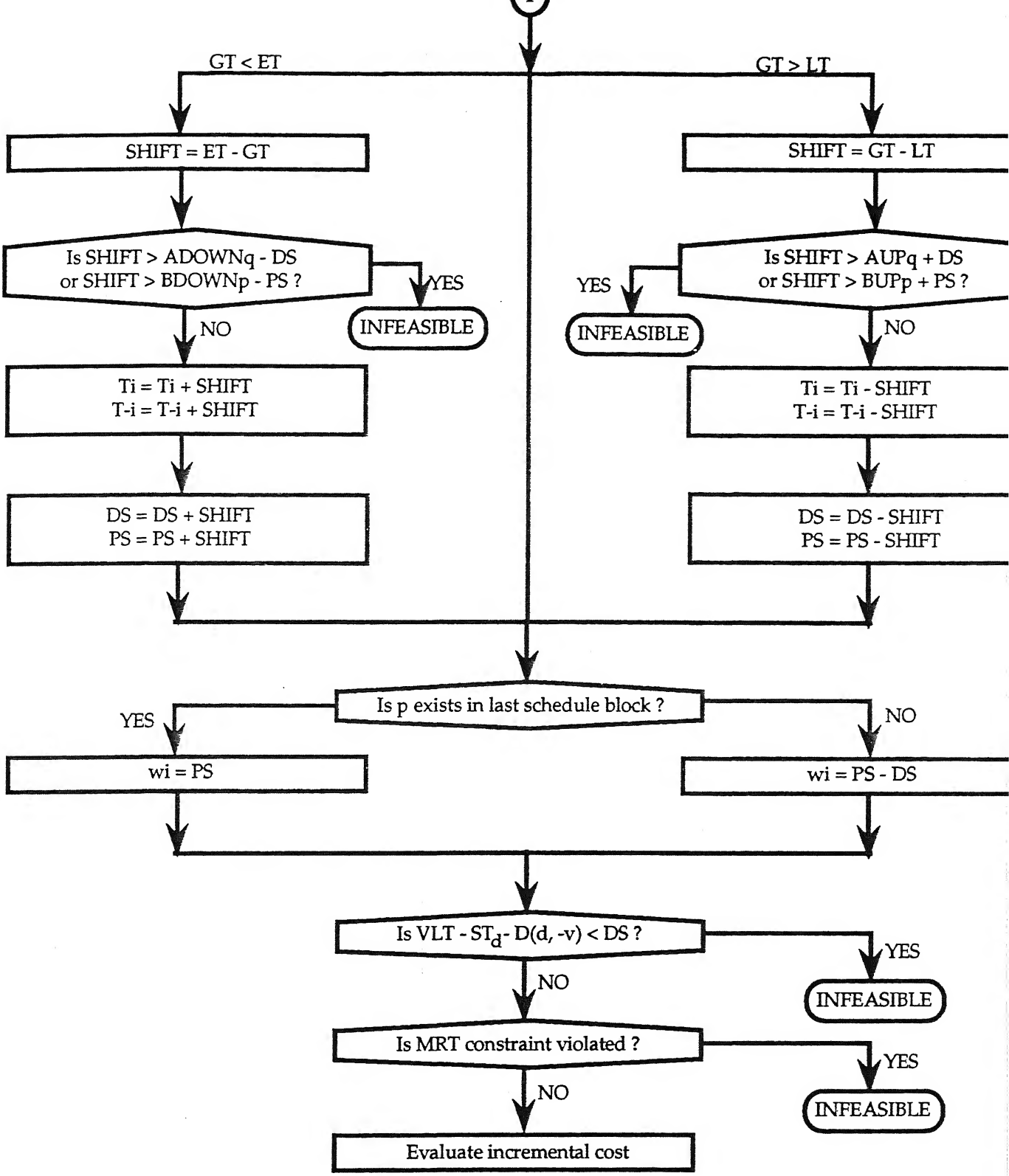

GT < ET
GT > LT
SHIFT = ET - GT
SHIFT = GT - LT
Is SHIFT > ADOWNq - DS
or SHIFT > BDOWNp - PS ?
YES
INFEASIBLE
NO
Is SHIFT > AUPq + DS
or SHIFT > BUPp + PS ?
YES
INFEASIBLE
NO
Ti = Ti + SHIFT
T-i = T-i + SHIFT
Ti = Ti - SHIFT
T-i = T-i - SHIFT
DS = DS + SHIFT
PS = PS + SHIFT
DS = DS - SHIFT
PS = PS - SHIFT
Is p exists in last schedule block ?
YES
NO
wi = PS
wi = PS - DS
Is VLT - ST_d - D(d, -v) < DS ?
YES
INFEASIBLE
NO
Is MRT constraint violated ?
YES
INFEASIBLE
NO
Evaluate incremental cost

Figure 3.9 (b)

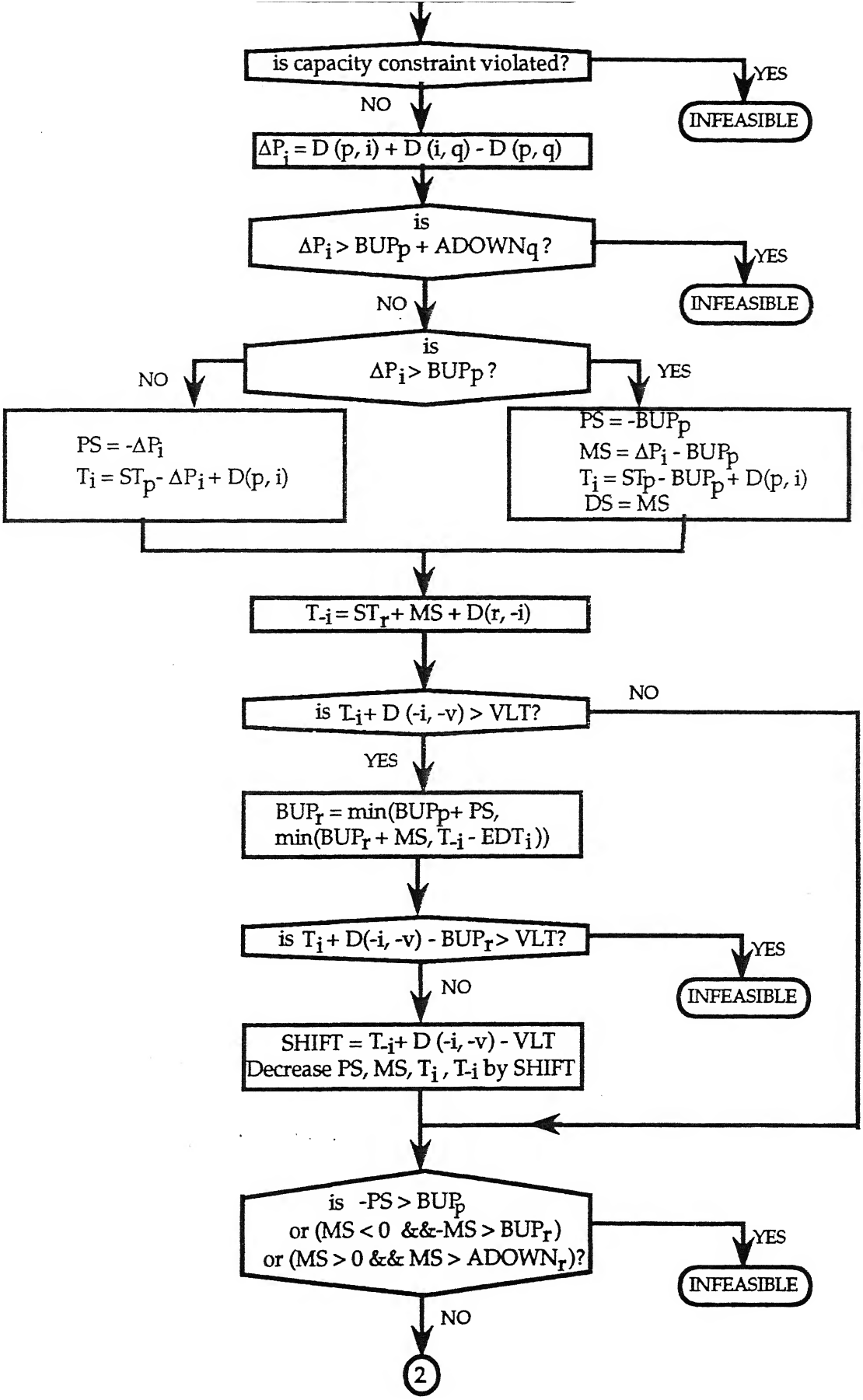

is capacity constraint violated?
YES
INFEASIBLE
NO
$\Delta P_i = D(p, i) + D(i, q) - D(p, q)$
is $\Delta P_i > BUP_p + ADOWN_q$?
YES
INFEASIBLE
NO
is $\Delta P_i > BUP_p$?
NO
YES
$PS = -\Delta P_i$
$T_i = ST_p - \Delta P_i + D(p, i)$
$PS = -BUP_p$
$MS = \Delta P_i - BUP_p$
$T_i = ST_p - BUP_p + D(p, i)$
$DS = MS$
$T_{-i} = ST_r + MS + D(r, -i)$
is $T_{-i} + D(-i, -v) > VLT$?
NO
YES
$BUP_r = \min(BUP_p + PS, \min(BUP_r + MS, T_{-i} - EDT_i))$
is $T_i + D(-i, -v) - BUP_r > VLT$?
YES
INFEASIBLE
NO
$SHIFT = T_{-i} + D(-i, -v) - VLT$
Decrease PS, MS, T_i, T_{-i} by SHIFT
is $-PS > BUP_p$
or $(MS < 0$ &&$-MS > BUP_r)$
or $(MS > 0$ && $MS > ADOWN_r)$?
YES
INFEASIBLE
NO
2

Figure 3.10 (a)

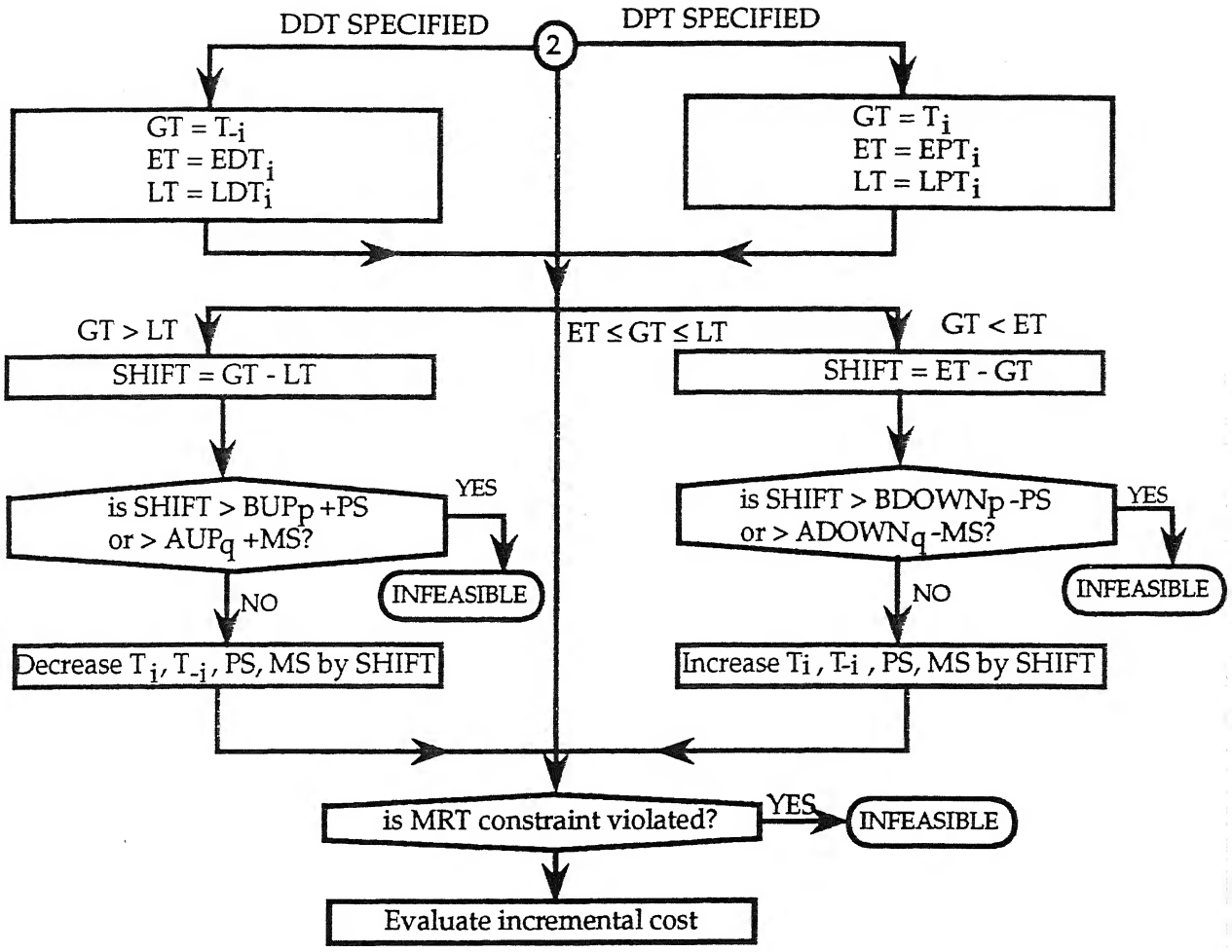
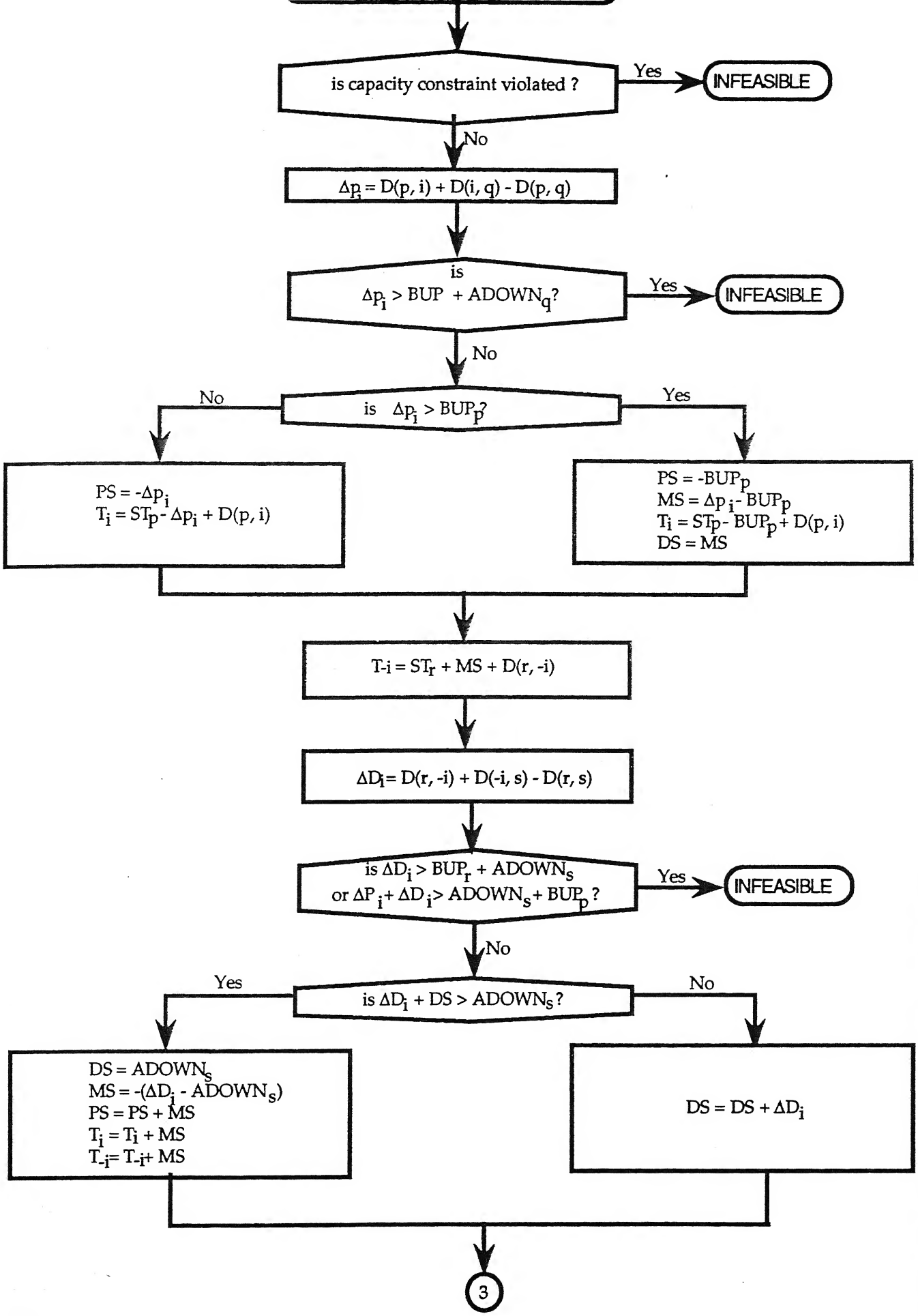
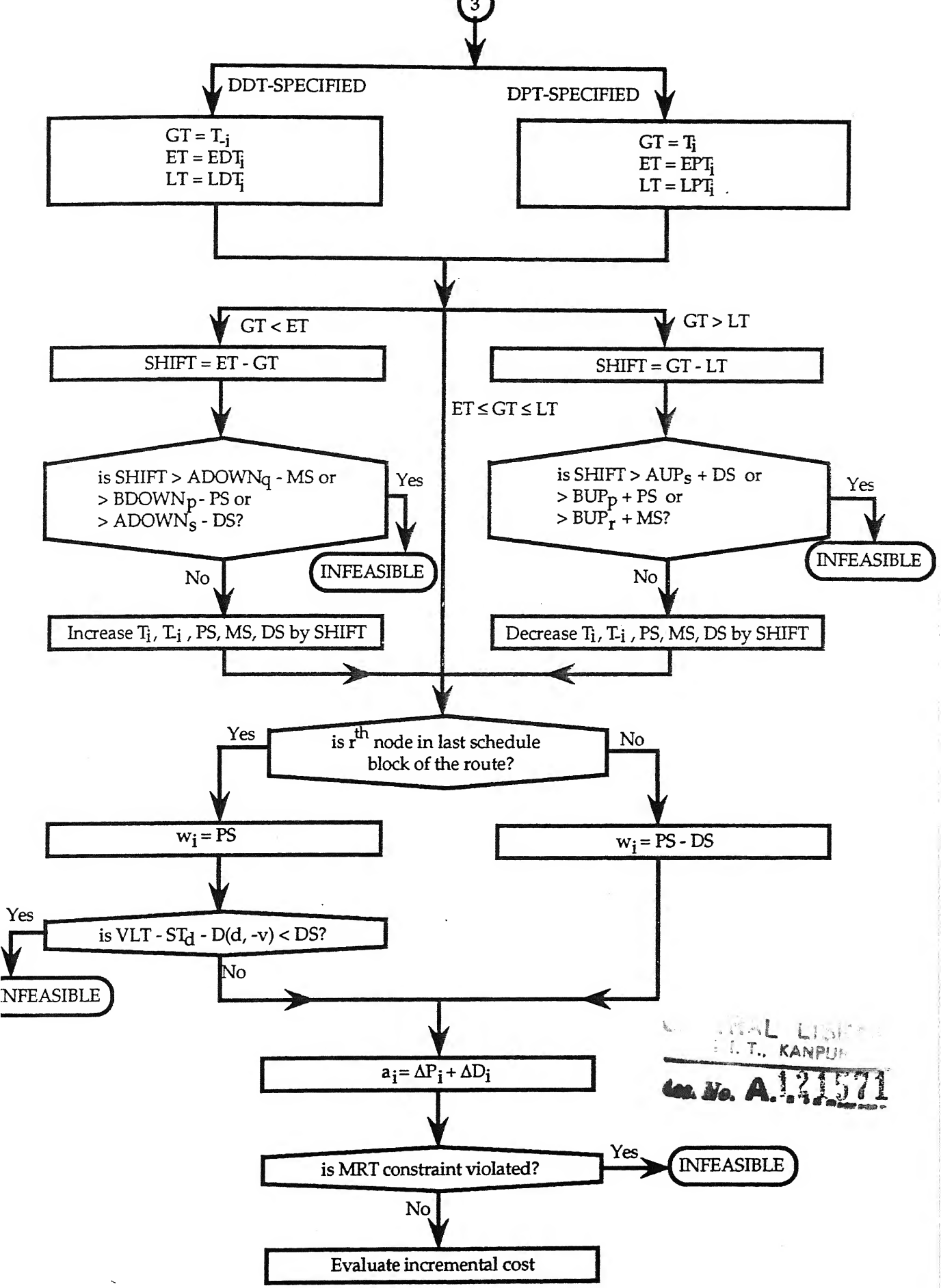

DDT SPECIFIED
2
DPT SPECIFIED
GT = T-i
ET = EDTi
LT = LDTi
GT = Ti
ET = EPTi
LT = LPTi
GT > LT
ET ≤ GT ≤ LT
GT < ET
SHIFT = GT - LT
SHIFT = ET - GT
is SHIFT > BUPp +PS
or > AUPq +MS?
YES
INFEASIBLE
NO
is SHIFT > BDOWNp -PS
or > ADOWNq -MS?
YES
INFEASIBLE
NO
Decrease Ti, T-i, PS, MS by SHIFT
Increase Ti , T-i , PS, MS by SHIFT
is MRT constraint violated?
YES
INFEASIBLE
Evaluate incremental cost

Figure 3.10 (b)


is capacity constraint violated ?
Yes
INFEASIBLE
No
$\Delta p_i = D(p, i) + D(i, q) - D(p, q)$
is $\Delta p_i > BUP + ADOWN_q$?
Yes
INFEASIBLE
No
No
is $\Delta p_i > BUP_p$?
Yes
$PS = -\Delta p_i$
$T_i = ST_p - \Delta p_i + D(p, i)$
$PS = -BUP_p$
$MS = \Delta p_i - BUP_p$
$T_i = ST_p - BUP_p + D(p, i)$
$DS = MS$
$T_{-i} = ST_r + MS + D(r, -i)$
$\Delta D_i = D(r, -i) + D(-i, s) - D(r, s)$
is $\Delta D_i > BUP_r + ADOWN_s$
or $\Delta P_i + \Delta D_i > ADOWN_s + BUP_p$?
Yes
INFEASIBLE
No
Yes
is $\Delta D_i + DS > ADOWN_s$?
No
$DS = ADOWN_s$
$MS = -(\Delta D_i - ADOWN_s)$
$PS = PS + MS$
$T_i = T_i + MS$
$T_{-i} = T_{-i} + MS$
$DS = DS + \Delta D_i$
3



SN_i : Slack time between the next schedule block and the block in which node i exists.

SP_i : Slack time between the previous schedule block and the block in which node i exists.

VLT : Latest time by which vehicle should reach its own destination or depot.

3.5.2 Feasibility Test for Insertions into Different Schedule Blocks

The insertion of the pick-up and delivery of customer i into different schedule blocks requires the estimation of all the slack time contained between these schedule blocks since, otherwise, the vehicle would wait idly with customer i on board. Let us consider the case of inserting the pick-up of customer i somewhere in schedule block k and the delivery of customer i into schedule block k+n ($n \geq 1$). We limit our search for feasible insertions to those that will not affect other customers on schedule blocks q, $q < k$ or $q > k+n$. Consequently, as a priori condition for an insertion to be considered in our search is :

$$\Delta P_i + \Delta D_i \leq \sum_{j=k}^{k+n+1} SLACK_j \qquad (3.19)$$

where ΔP_i (ΔD_i) is the extra vehicle time required to serve the pick-up (delivery) point of customer i. This is shown in the figure 3.12.

The algorithmic approach works as follows :

Form a new schedule sequence by merging schedule blocks k to k+n and with customer i inserted. Let d denote the number of stops in the new

schedule sequence. Construct an earliest time schedule (without considering the time constraints at each stop) for the new sequence with all the slack time eliminated, i.e., by taking $SLACK_k = 0, SLACK_{k+1} = 0, \ldots, SLACK_{k+n} = 0$. Let us denote this tentative time schedule by "T." For every stop r (r = 1, 2, ..., d) in the new schedule sequence, compute two statistics, R_r and A_r :

$$R_r = |\text{Min}\,(T_r - ET_r, 0)| \tag{3.20}$$

$$A_r = LT_r - T_r \tag{3.21}$$

where T_r is the visiting time of stop r in time schedule T.

R_r denotes the minimum time by which the visit of stoop r, T_r, should be delayed for it to fall within the time window of stop r. A_r denotes the maximum time by which T_r can be delayed for it not to violate the time window constraint of stop r. After R_r and A_r are computed for all r in the new schedule sequence, we can define R_{min} and A_{max} for time schedule T as :

$$R_{min} = \underset{\text{all r}}{\text{Max}}\ (R_r) \tag{3.22}$$

$$A_{max} = \underset{\text{all r}}{\text{Min}}\ (A_r) \tag{3.23}$$

R_{min} represents the minimum amount of time by which the time schedule T should be delayed so that every stop in T is visited at or after its earliest feasible time. A_{max} represents the maximum amount of time by which the time schedule T can be delayed without a stoop being visited later than its latest feasible time. For a schedule sequence to be feasible we must have :

$$R_{min} \leq A_{max} \tag{3.24}$$

(3.24) indicates the fact that the amount of time by which T has to be delayed must be less than or equal to the maximum amount of time that T can be delayed. (3.24) is the necessary and sufficient condition for a new sequence to be feasible with respect to satisfying the time window constraints. To check

for violations of vehicle capacity and customer's maximum-ride-time constraints, a screening test through the new time schedule would suffice. (Such screening can be performed at the same time when R_r and A_r computed. See Figure 3.12)

We now proceed to discuss the optimization procedure used to choose the most desirable insertion.

3.6 THE OPTIMIZATION PROCEDURE

In order to select all feasible insertions of customer i into the work-schedules of the available vehicles, we use an objective function designed to evaluate the incremental "cost" of each insertion. This cost is taken to be a weighted sum of disutility of system's customers and of operator costs - the latter measured in terms of "consumption" of available vehicle resources.

Assume that it is feasible to insert customer i into the work-schedule of vehicle j. Then the incremental disutility of that insertion to the system's customers consists of the sum of two parts: the disutility of customer i, i.e., the customer being assigned to a vehicle; and the additional disutility suffered by all *other* customers *already assigned to that vehicle* because of the insertion of customer i. The first part (disutility of customer i) is given by

$$DU_i = DU_i^d + DU_i^T \tag{3.27}$$

where DU_i^d = disutility due to deviation from most desired time

$$= C_1 x_i + C_1 x_i^2 \qquad 0 \le x_i \le WS_i \tag{3.28}$$

where WS_i is defined in section 3.3.

and,

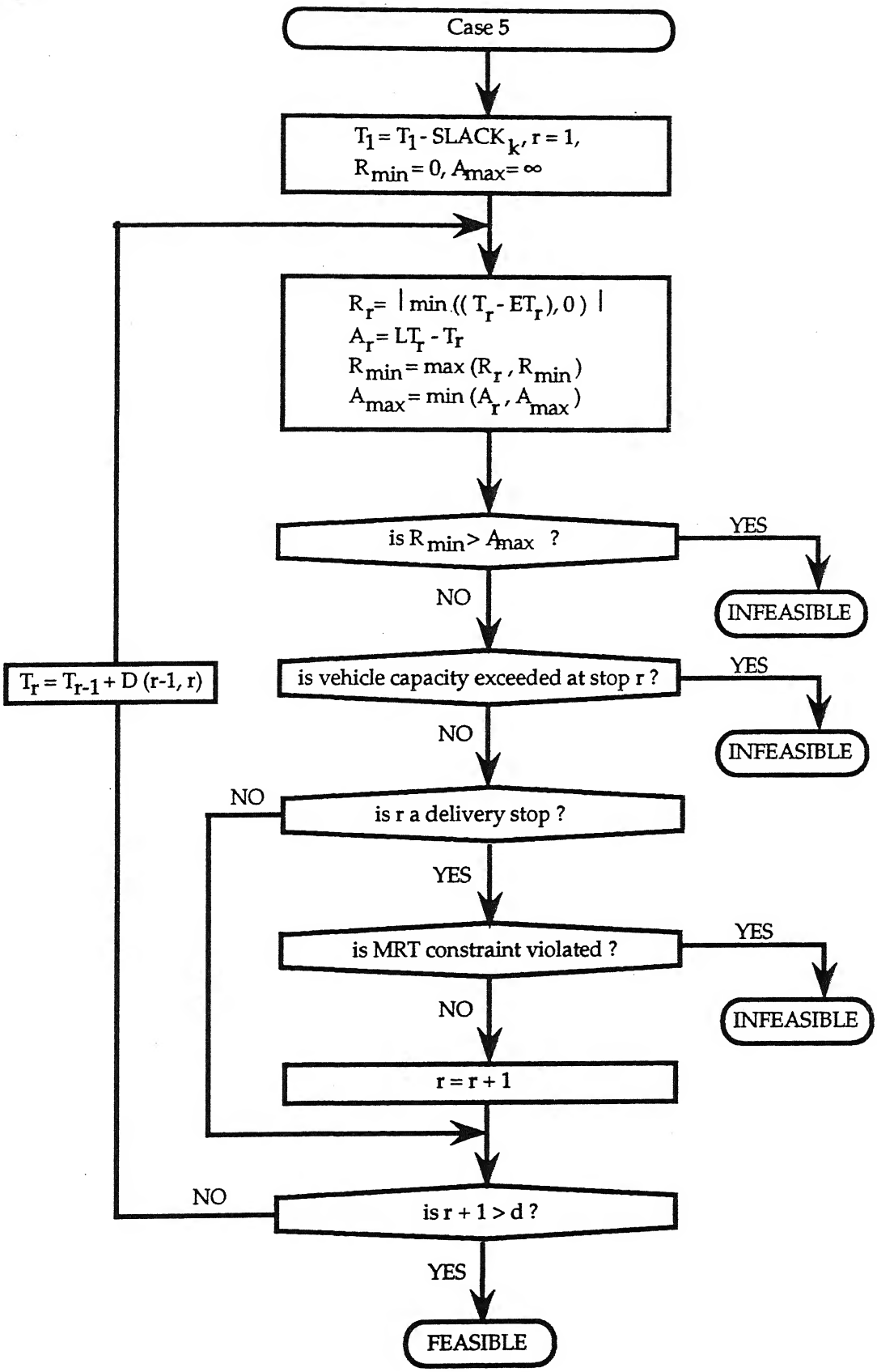

Case 5
$T_1 = T_1 - SLACK_k$, r = 1,
$R_{min} = 0$, $A_{max} = \infty$
$R_r = | \min ((T_r - ET_r), 0) |$
$A_r = LT_r - T_r$
$R_{min} = \max (R_r, R_{min})$
$A_{max} = \min (A_r, A_{max})$
is $R_{min} > A_{max}$?
YES
INFEASIBLE
NO
$T_r = T_{r-1} + D(r-1, r)$
is vehicle capacity exceeded at stop r ?
YES
INFEASIBLE
NO
NO
is r a delivery stop ?
YES
is MRT constraint violated ?
YES
INFEASIBLE
NO
r = r + 1
NO
is r + 1 > d ?
YES
FEASIBLE

Figure 3.12

DU_i^T = disutility due to excess ride time

$$= C_3 y_i + C_4 y_i^2 \qquad y_i \geq 0 \tag{3.29}$$

In (3.28) and (3.29), C_1, C_2, C_3 and C_4 are user-specified constants and

$$x_i = \begin{cases} APT_i - DPT_i & \text{for DPT-specified customers} \\ DDT_i - ADT_i & \text{for DDT-specified customers} \end{cases} \tag{3.30}$$

and $$y_i = ART_i - DRT_i \tag{3.31}$$

The quadratic terms allow the modeling of situations in which disutilities are believed to increase nonlinearly with x_i and/or y_i. Clearly, by varying C_1, C_2, C_3 and C_4 (including assigning the value of zero to some of them) many different types of behavior can be represented.

The second part (additional disutility to other customers) is given by :

$$DU_i^0 = \sum_{k \text{ on vehicle } j} (DU_k^{new} - DU_k^{old}) \tag{3.32}$$

where DU_k^{new} and DU_k^{old} are, respectively, the disutilities to customer k after and before the insertion of customer i into the schedule of vehicle j. The summation is over all customer k who were already assigned to vehicle j prior to the assignment of customer i.

The incremental cost, VC_i, to the system's operator due to inserting customer i into the work-schedule of some vehicle is quantified in somewhat unusual terms by our objective function. We have,

$$VC_i = C_5 z_i + C_6 w_i + U_i(C_7 x_i + C_8 w_i) \tag{3.33}$$

where C_5, C_6, C_7 and C_8 are user defined constants.

z_i is the additional active vehicle time required to serve the customer i

w_i is the change in vehicle slack time due to the insertion

U_i is an indicator of system workload defined as :

$$U_i = \frac{\text{(No. of system customers in } [EPT_i - W_1,\ EPT_i + W_2])}{\text{(Effective no. of vehicles available in } [EPT_i - W_1,\ EPT_i + W_2])} \qquad \ldots \quad (3.34)$$

In (3.34), W_1 and W_2 are user specified constants. For example, if $W_1 = 0, W_2 = 60$ minutes, U_i is equal to the ratio of the number of customers demanding service to the available number of vehicles during the one-hour time period that has the earliest pick-up time of customer i as its mid-point. The word "effective" is used in the denominator of (3.34) to account for the fact that some vehicles may be available for service only for a fraction of the time interval (e.g., two hours) in question - usually because of driver constraints, union rules, etc.

The following should be noted with respect to (3.33) :

(i) The change in vehicle slack time, w_i, can be positive or negative. It will be negative if the insertion means that slack time in the original schedule will now be utilized to serve customer i; it will be positive if, in order to serve customer i, additional vehicle slack time must be created (this will happen if a new schedule block is created to serve customer i).

(ii) In practice, the cost per unit of time of a vehicle in the "active" state is greater than in the "slack" state (e.g., no fuel consumption in slack state). Therefore, we must have $C_5 \geq C_6$ and $C_7 \geq C_8$.

(iii) Obviously U_i will be larger during periods of heavy demand. Since the total "cost" of an insertion is given by $DU_i + VC_i$ - i.e. by the sum of (3.27), (3.32) and (3.33) - it is clear that during heavy demand periods,

the objective function places more emphasis on the system operator's costs (relative to customers' disutility) than during low demand periods. This is as it should be, since during periods of high utilization vehicle resources are scarcer and it is thus important to "conserve" these resources as much as possible. For all test runs of ADARTW to be described in Chapter 5, $W_1 = 0$ and $W_2 = 60$ minutes are chosen to compute U_i in (3.34). U_i in this case can be explained as the average number of customers a vehicle would have to carry in the next hour starting from the earliest pick-up time of customer i. As a result, the objective function in the optimization step which includes U_i as a parameter will take into consideration the customer demands for the next hour and adjust automatically the weight placed on the vehicle resource term. In the last hour of dial-a-ride operation U_i will become smaller as we approach the end of the subscription list. This is a desirable feature since at the end of a day no urgent conservation of vehicle resources is necessary.

Given this background, the optimization steps of ADARTW are given in figure 3.13.

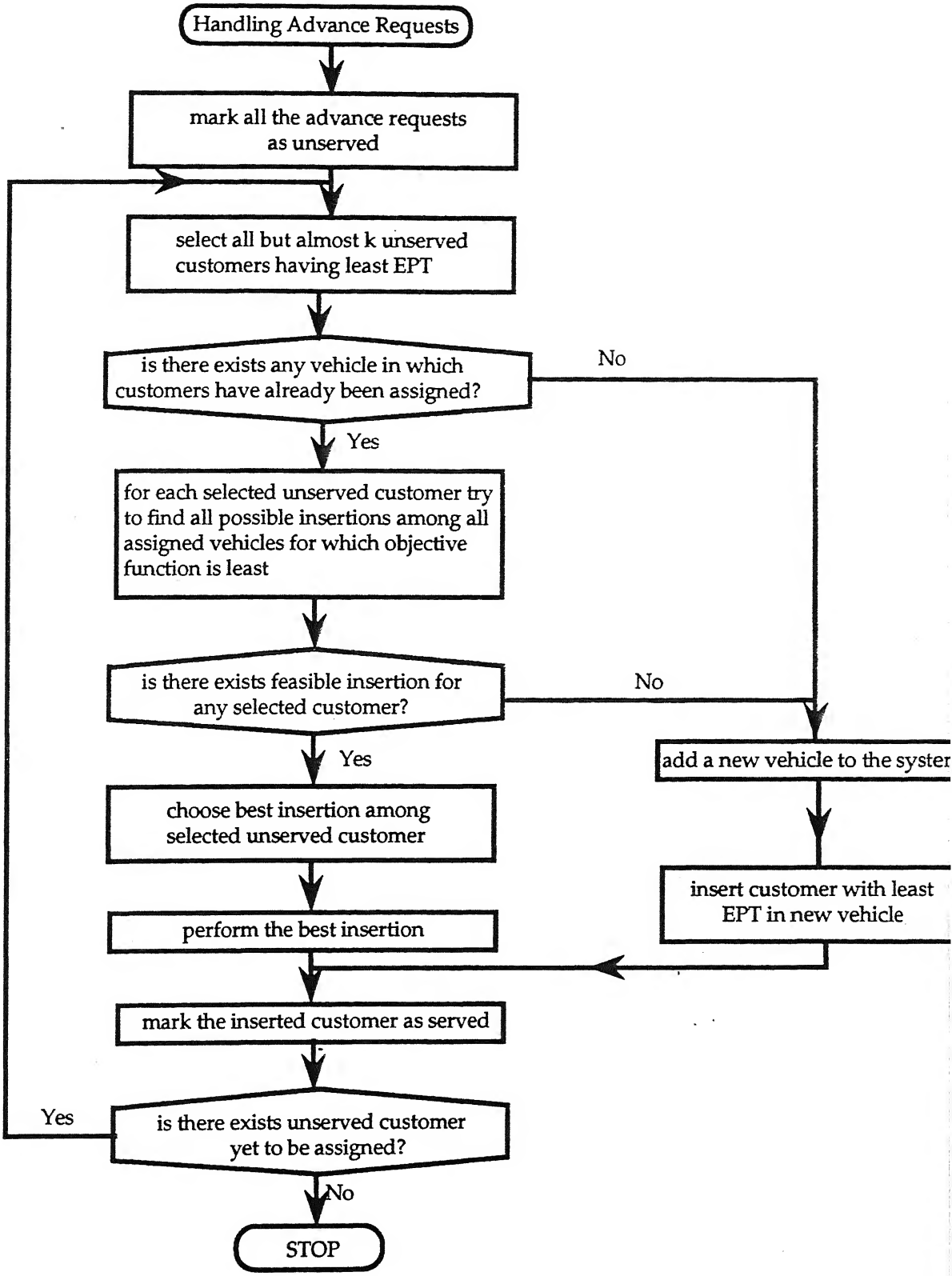

Handling Advance Requests
mark all the advance requests as unserved
select all but almost k unserved customers having least EPT
is there exists any vehicle in which customers have already been assigned?
No
Yes
for each selected unserved customer try to find all possible insertions among all assigned vehicles for which objective function is least
is there exists feasible insertion for any selected customer?
No
Yes
add a new vehicle to the syster
choose best insertion among selected unserved customer
insert customer with least EPT in new vehicle
perform the best insertion
mark the inserted customer as served
Yes
is there exists unserved customer yet to be assigned?
No
STOP

Figure 3.13

3.7 COMPUTATIONAL RESULTS AND ANALYSIS

3.7.1 Introduction

In order to gain insights into the performance of ADARTW algorithm we generated the necessary input data. The following set of information is required as an input to the software developed for ADARTW algorithms:

Time matrix

The direct ride time between all the nodes, which will act as either origin or destination for the customers, is stored in the time matrix.

Customer subscription lists

This is the necessary input specified by the customers. We have generated a set of customers subscription lists. These lists are based upon combinations of different demand scenarios. By different demand scenario we mean that number of customers, requesting for the service per hour, are not constant and it may vary from 10 customers per hour to 100 customers per hour. We have maintained two types of service quality levels that could be guaranteed by the system operators. These service quality levels will be differentiated according to the guarantees provided by system operator to their customers in terms of time window and ride time constraints.

The constituents of the customer subscription lists includes the following :

Desired pickup or delivery time.

Service quality required.

Number of persons associated with a request.

Origin.

Destination.

Information regarding vehicles

This information is specified by the system operator. This include the following :

Origin of vehicle.

Destination of vehicle.

Capacity of the vehicle.

Availability times of a vehicle during a day.

In the following section, we will describe in detail that how we have generated these inputs.

3.7.2 Input Data

We have assumed that vehicle will be serving an area of 6 x 6 square miles. We have identified 100 different points which will serve as the origins and destinations for the different customers. We have plotted these points by hand such that they are evenly distributed in the service area.

We have generated the time matrix by dividing the distance between the nodes by the average speed of the vehicle. Average speed of the vehicles are assumed to be 15 miles per hour. Here distances assumed to be rectilinear. Rectilinear distance between two points $i(x_i, y_i)$ and $j(x_j, y_j)$ is given by following expression :

Rectilinear distance $= |x_i - x_j| + |y_i - y_j|$

Vehicle depot is assumed to be at the center of the service area. All the vehicles will start from the vehicle depot and at the end of a day they will return back to the depot. However, there exists the flexibility that the origin and destination of each vehicle can be defined distinctly by the system operator. Each vehicle may have different capacities, but for our experimentation we have assumed it to be eight. We have also assumed that the vehicles are available all the 24 hours, however there exists the flexibility that for each vehicle, system operator may define the availability periods distinctly. For example system operator may say that this vehicle is available during the period from 8 a.m. to 12 a.m. and then from 2 p.m. to 6 p.m. Information regarding vehicles are generated keeping the above things in mind.

While generating subscription lists for customers, we have restricted the duration in which customers can ask for the service between 9 a.m. to 5 p.m. In order to spread the requested service timings evenly throughout the duration in which service is provided, we have divided the service duration (from 9 AM to 5 PM) into 4 equal parts, each of two hour duration, i.e., [9 AM, 11 AM), [11 AM, 1 PM], [1 PM, 3 PM] and [3 PM, 5 PM]. For each instance of the customer subscription list 25% of the desired pickup or delivery time of the customer will fall into one of time slots. In each time slot the request times are generated randomly. Similarly, the origin and destination points of the customers are distributed evenly throughout the service area.

Percent of DPT specified customers are kept to 50% in all the instance generated. Also, percent of customers demanding expensive service are kept to 50%. For each request we have assumed the number of persons associated with it will have the following probability distribution :

Number of person 1 probability 0.7.

Number of person 2 probability 0.2.

Number of person 3 probability 0.1.

For the two different service quality levels assumed, the difference exists in the following constraints :

Time window size : 20 minutes for expensive service

30 minutes for normal service

Maximum Ride Time : 10 + 1.5 DRT for expensive service

10 + 1.7 DRT for normal service.

where DRT (direct ride time) is in minutes.

Parameters W_1 and W_2 are assumed to be $W_1 = 0$, $W_2 = 60$ minutes.

With the above assumptions we have prepared the subscription list of different sizes. These are 100, 200, 300, 400, 500 and 1000 customers in each subscription list. In turn we have generated 5 instances of each size.

3.7.3 Computational Results and Analysis

We divide the discussion of computational results on simulated data into two parts. The first part focuses on the question of how different parameters (C_1, C_2 and others) in the objective function affect the results of ADARTW.

The second part examines the performance of different pool refilling strategies.

3.7.4 Investigation of the effects of individual parameters

In the first part of the investigations, we use $C_1, C_2, C_3, C_4, C_5, C_6 = 0$, $C_7 = 1.2$, $C_8 = 0.7$ as the **base case** parameter set. By varying one parameter at a time in

the subsequent test runs, we attempt to separate the effects of the individual parameter. We have taken $C_7 > C_8$ because of the reasons explained in section 3.6.

We have collected the following statistics at the end of each run :

- Average Deviation $= \frac{1}{N} \sum_{i=1}^{N} (APT_i - EPT_i)$ for DPT-specified customers

  $= \frac{1}{N} \sum_{i=1}^{N} (LDT_i - ADT_i)$ for DDT-specified customers

- Average Ride Time Relation $= \frac{1}{N} \sum_{i=1}^{N} \frac{ADT_i - APT_i}{DRT_i}$

- Vehicle Productivity $= \frac{N}{\text{Total Vehicle Time}}$

- Number of Vehicles used

- Vehicle Utilization Rate $= \frac{\text{Active Vehicle Time}}{\text{Total Vehicle Time}}$

- Total Slack

We have chosen the instance of 500 request id for our investigation purpose. We have assumed that initially we have 100 vehicles available with us.

(i) Parameter C_1

In this experiment it is assumed that the customer disutility function is a linear function of service time deviation, i.e., DU_i^d can be expressed as $DU_i^d = C_i x_i$ as $C_i = 0$ for i = 2 ,3, 4, 5 and 6 in objective function. To investigate the effect of different values of C_1 on the solution of ADARTW, a series of test runs were conducted by varying the value of C_1 from the **base case** scenario in each successive run.

We have plotted graphs for some important statistics to observe the effect of variation in C_1.

As the value of C_1 is increased, as expected, the average deviation from desired time is reduced. As shown in figure 3.14, average deviation reduces from 11.5 minutes for value of C_1 = 0.1 to 5.5 minutes. This trend is consistent with the purpose for which this parameter was conceived in objective function.

The effect on excess ride time can be seen from the figure 3.15. Although no customer disutility was assumed for excess ride time in this test run ($C_3 = C_4 = 0$), the average ride time ratio actually reduces from 1.46 to 1.32 with the increase in the value of C_1. One explanation for such behavior is that as more weight is placed on time deviation, customers are likely to be delivered directly from their origin to their destination. so that their service time are close to their desired time. Another reason might be that as the value of C_1 increases more vehicles are introduced into the system and consequently the system will provide better ride service when the workload is shared by additional vehicle.

Another important statistic active and total vehicle time required is plotted in the figure 3.16. When no customer disutility was specified in **base case** Total vehicle time required by all the 31 vehicles was 15000 minutes. The quality of service provided can be evaluated by an average deviation of 11.5 minutes from a customer's desired pickup or delivery time and by an average 47% more ride time than necessary. As C_1 increases quality of service improves and total time required by vehicle is also increased to approximately 16200 minutes. As expected figure 3.17 shows that number of vehicles required are also increased with the increase in the value of C_1.

Effect of varying parameter C1 on deviation from desired service time (pickup or delivery)

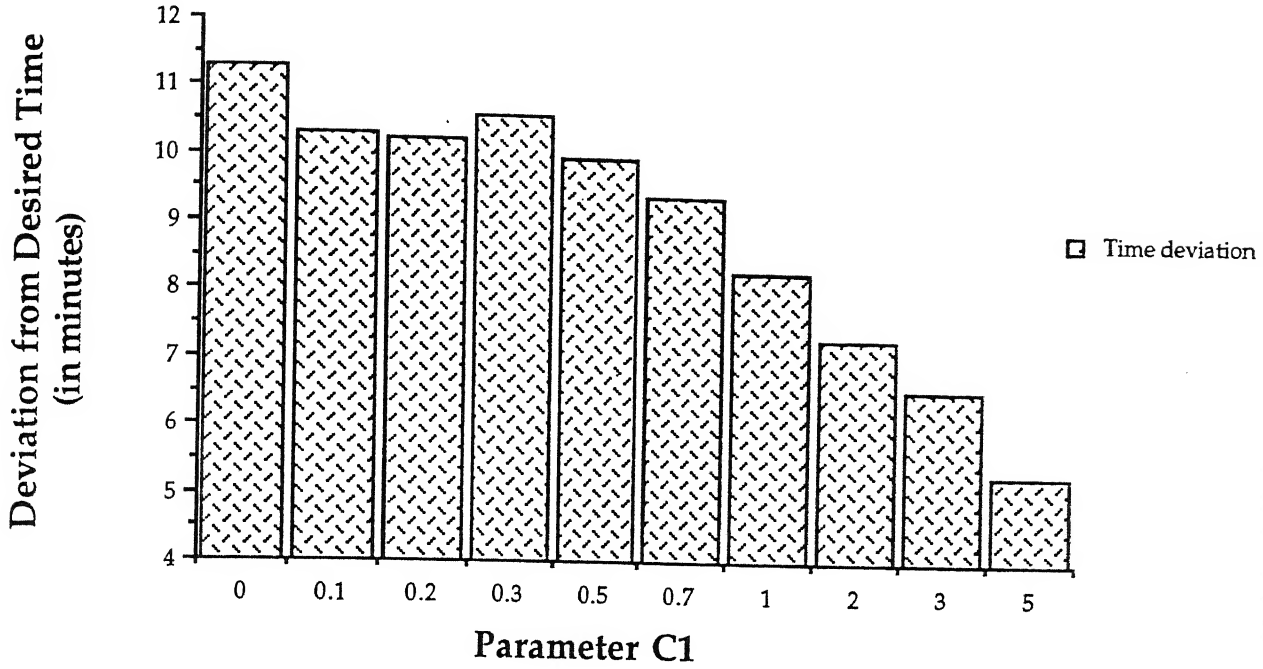


Figure 3.14

Effect of varying parameter C1 on ride time ratio

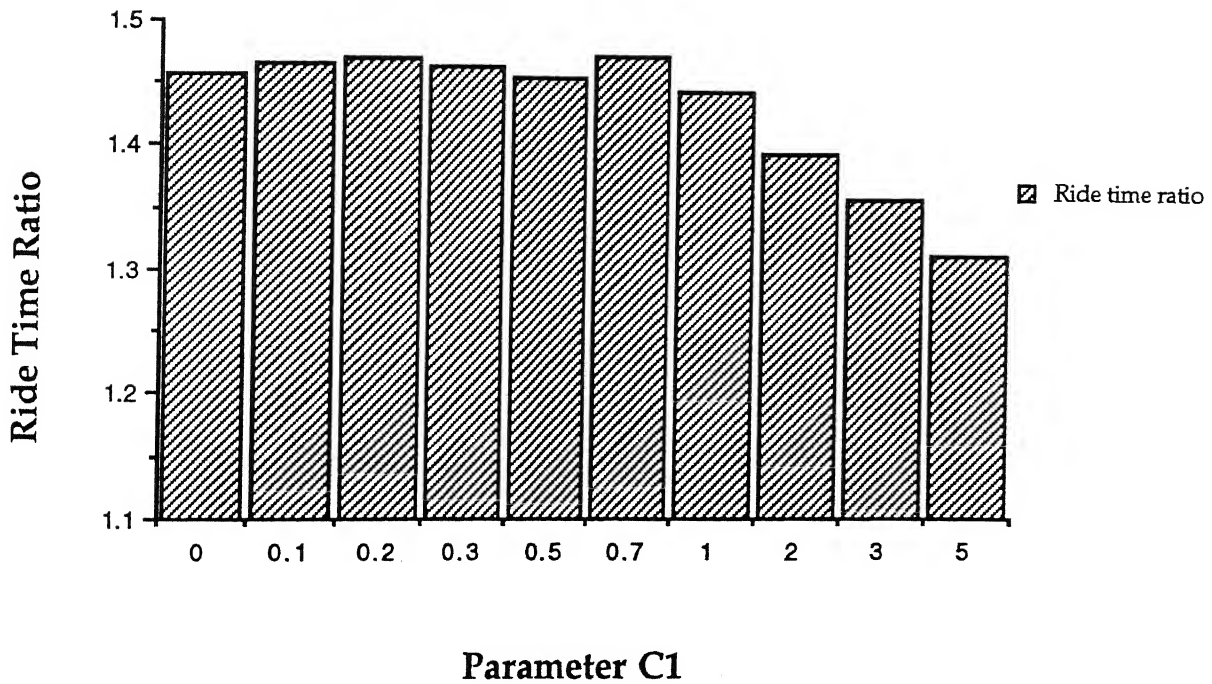


Figure 3.15

Effect of varying parameter C1 on active vehicle time and total vehicle time

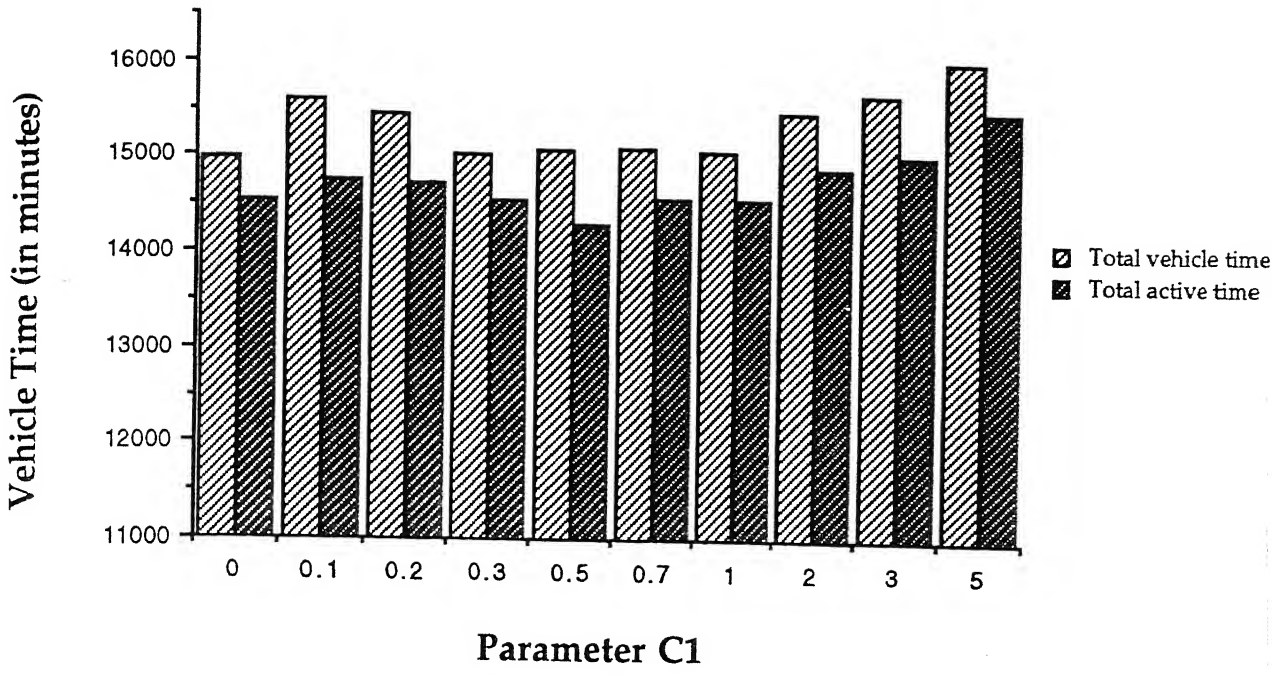


Figure 3.16

Effect of varying parameter C1 on number of vehicles used

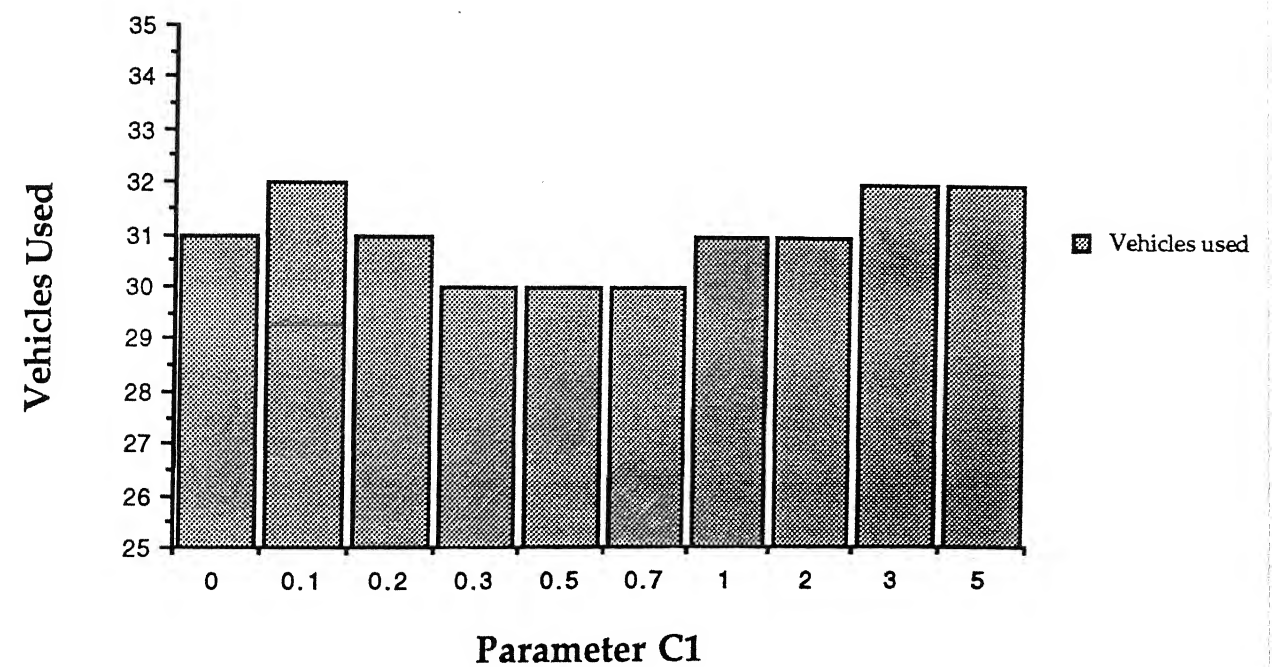


Figure 3.17

Effect of varying parameter C1 on vehicle productivity

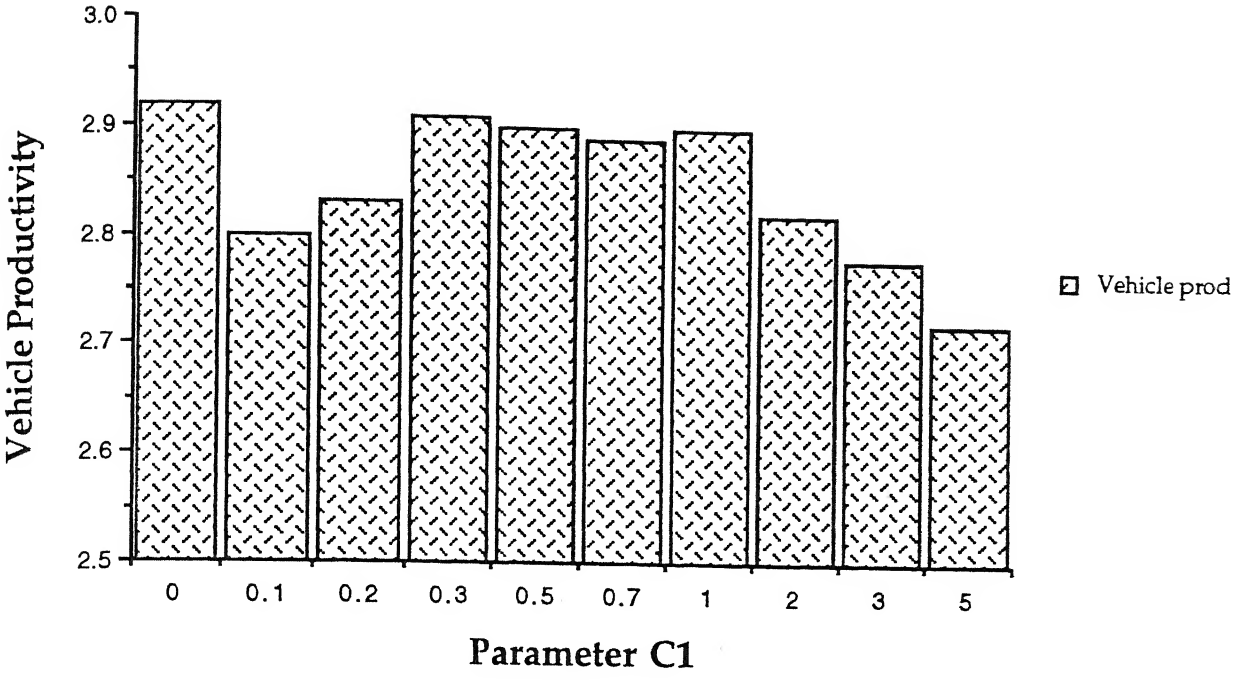


Figure 3.18

Effect of varying parameter C1 on vehicle utility

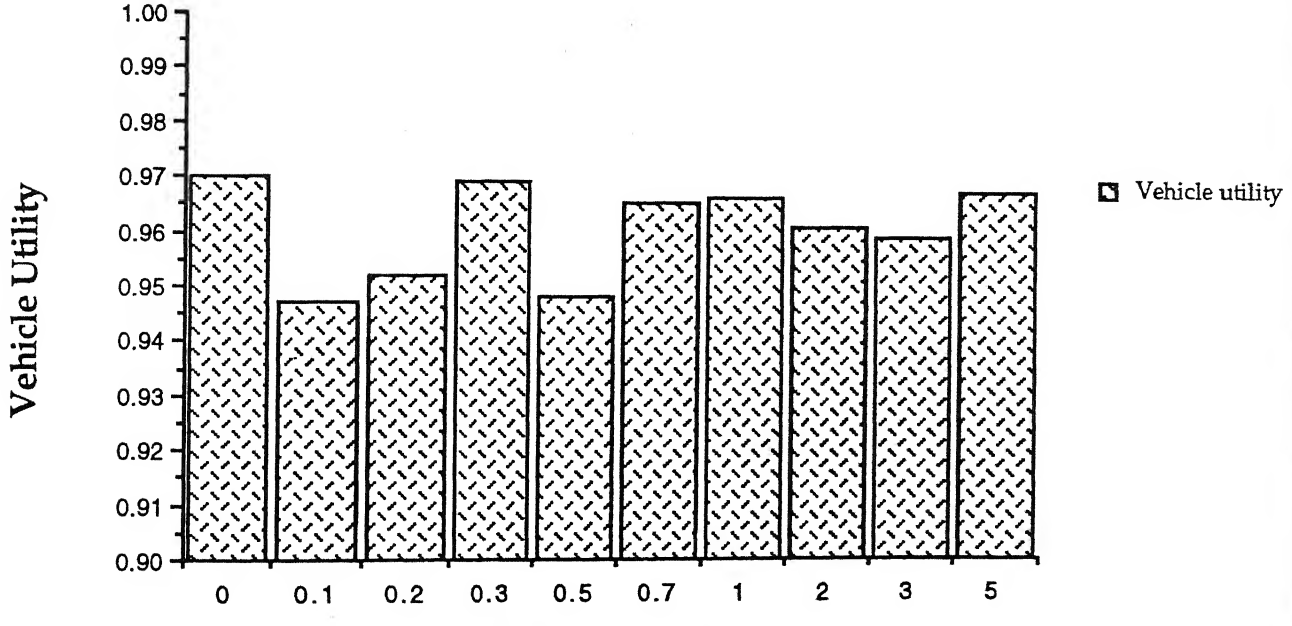


Figure 3.19

Vehicle productivity dips from 2.92 for the base case to 2.73 as displayed in the figure 3.18. Vehicle utilization remains in the range of 95% which can be observed from the figure 3.19. This is because of the fact that increase in the active vehicle time also results in the increase in total vehicle time as shown in figure 3.16.

It is hard to predict the best value of parameters because of "myopic" nature of ADARTW. But the global trend between quality of solution and parameter C_1 is obvious and can be predicted in long run.

(ii) **Parameter C_2 :**

Here we assume that disutility function, DU_i^d, is quadratic in the service time deviation : $DU_i^d = C_2 x_i^2$. Similar experiment runs were designed as for C_1.

The trends in the statistics are consistent with our observation during previous investigations of C_1.

Here there is a sharp decline in the deviation from desired time which in base case was 11.5 minutes and for $C_2 = 1$ it is 5.5 minutes as shown in the figure 3.20. To achieve same deviation the value of C_1 required was 5. This is expected since the disutility function is quadratic in nature.

As shown in the figure 3.21 excess ride time is reduced by 13% from 1.46 to 1.33 for base case and for $C_2 = 1$ respectively.

The requirement for vehicle resources also rises steeply in this case because of the quadratic disutility function. As shown in figure 3.16 and figure 3.22 although for $C_1 = 5$ and $C_2 = 0.7$ quality of the service (ride time ratio and deviation) is almost same and active vehicle time for $C_1 = 5$ was less by 400 minutes but number of vehicles required are less by 4 (figure 3.17 and figure 3.23). This means that addition of the new vehicle to the system is taking place earlier to satisfy the high service quality constraint.

Effect of varying parameter C2 on deviation from desired service time (pickup or delivery)

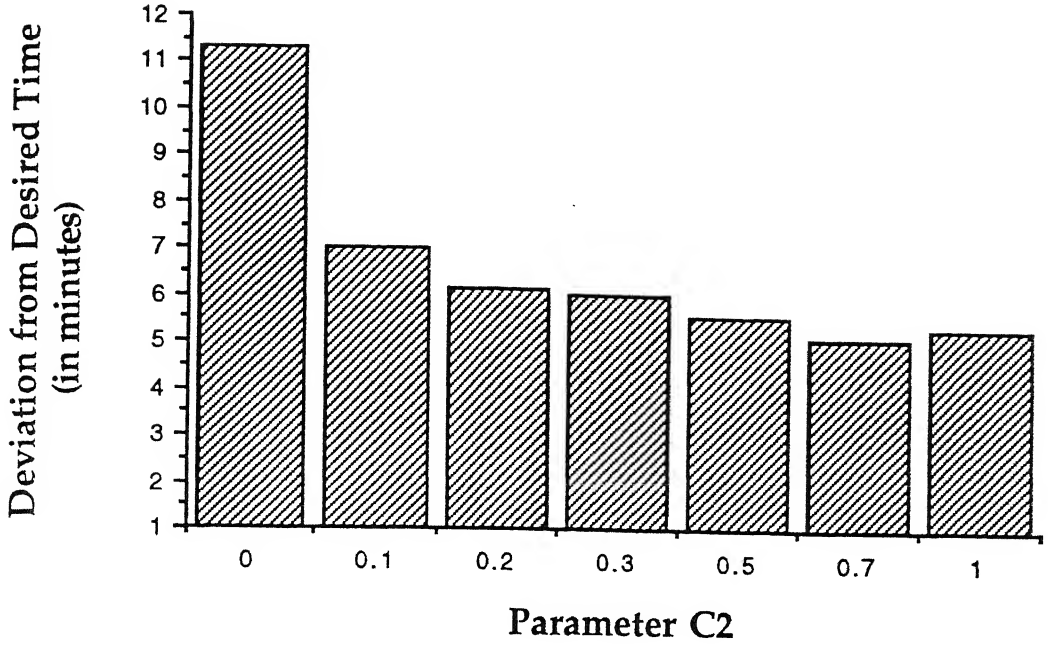


Figure 3.20

Effect of varying parameter C2 on ride time ratio

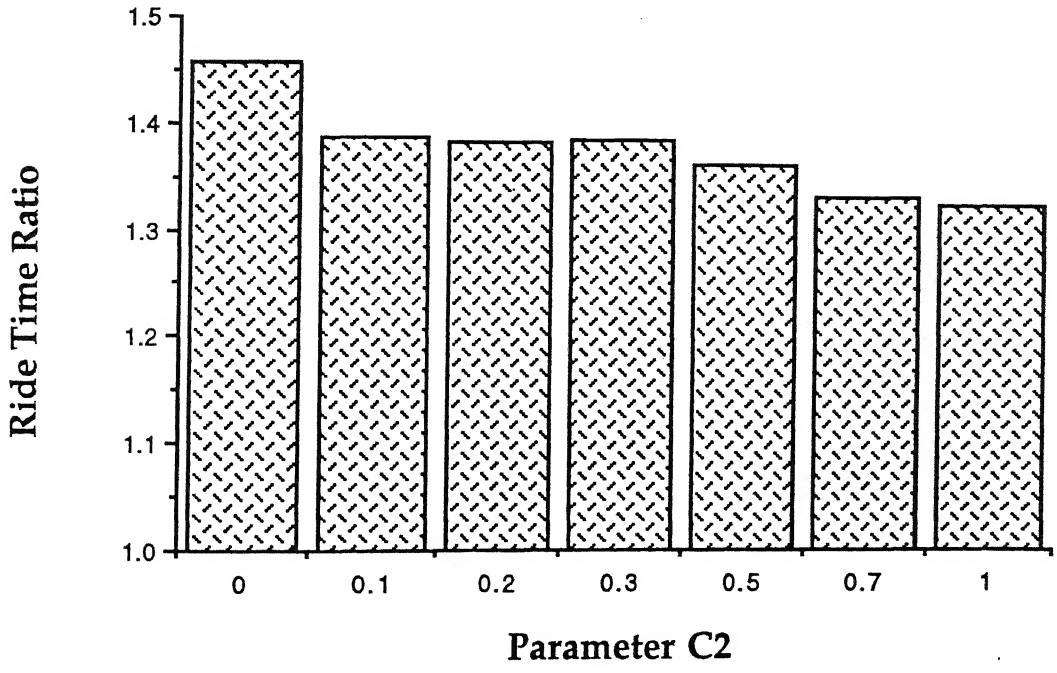


Figure 3.21

Effect of varying parameter C2 on active vehicle time and total vehicle time

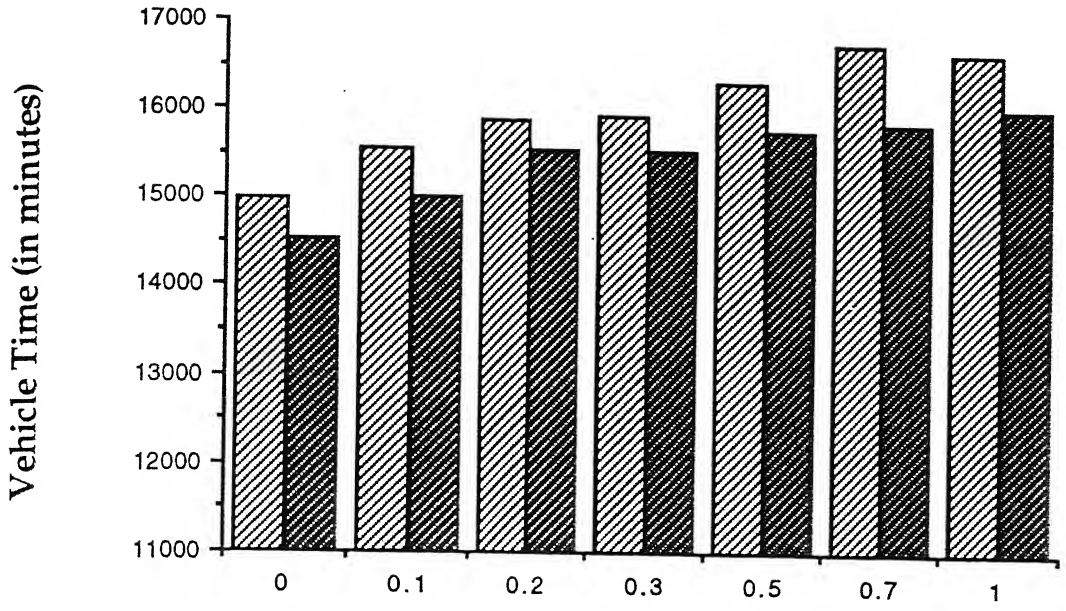


Figure 3.22

Effect of varying parameter C2 on number of vehicles used

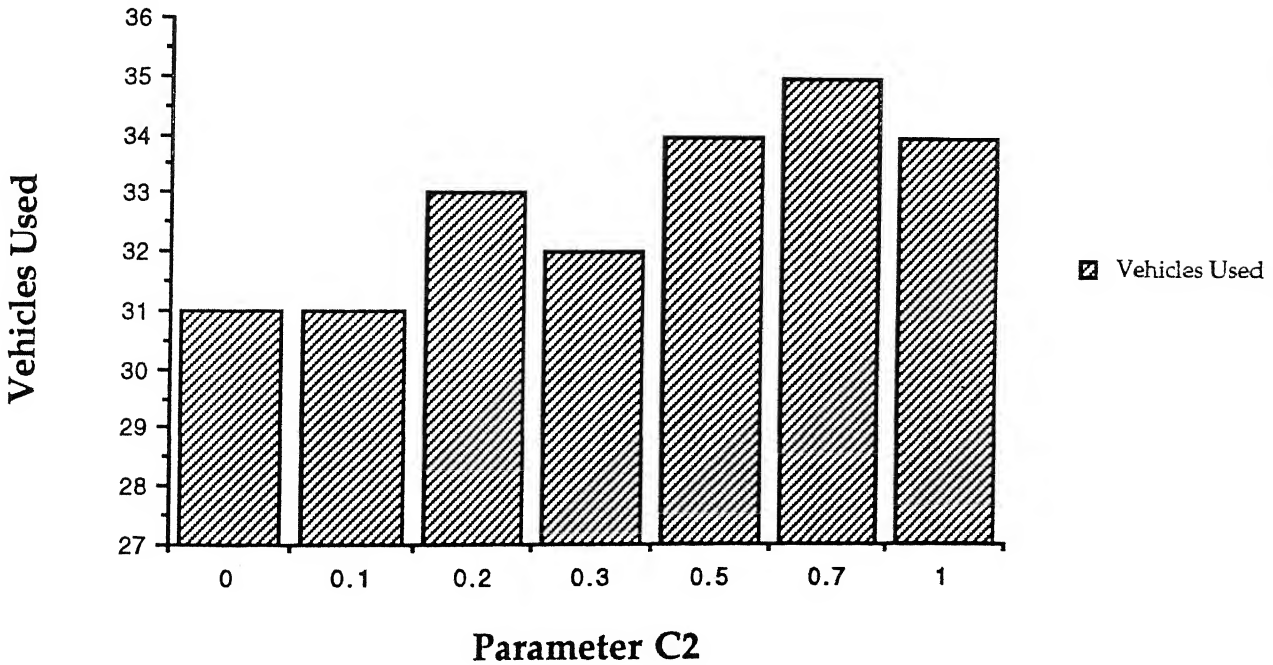


Figure 3.23

Effect of varying parameter C2 on vehicle utility

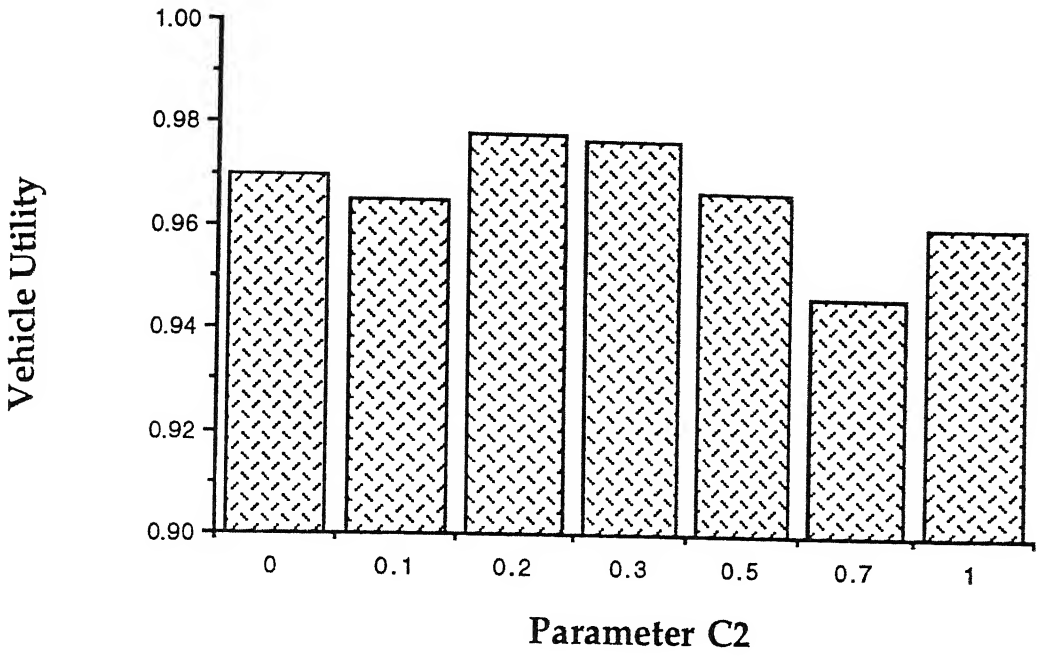


Figure 3.24

Effect of varying parameter C2 on vehicle productivity

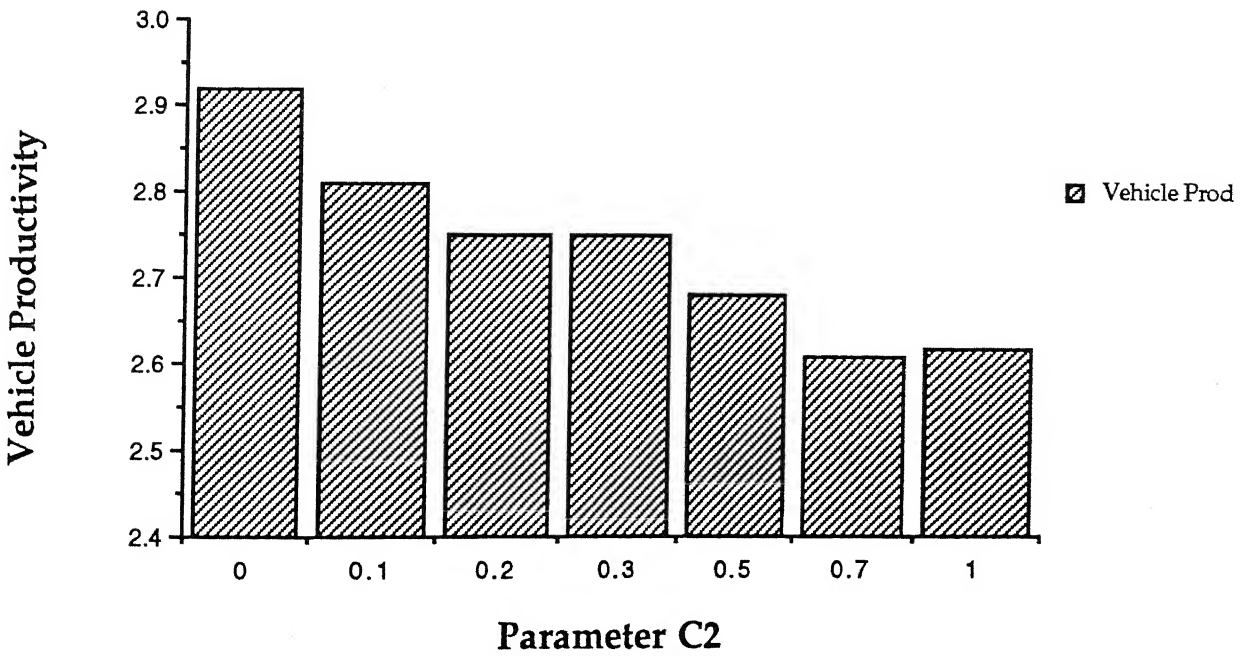


Figure 3.25

Graph displaying the distribution of time deviation for all the customers

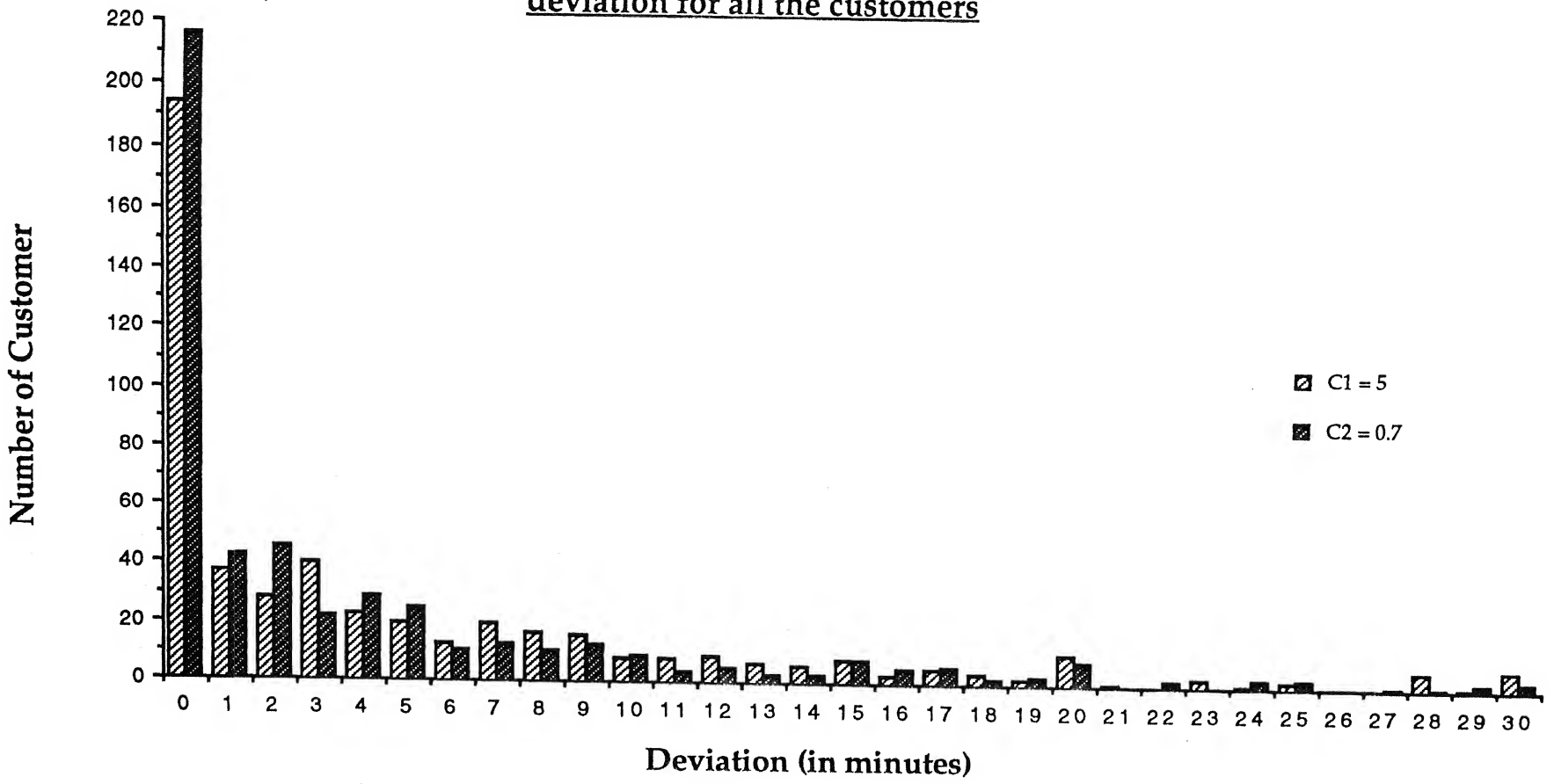


Figure 3.26

Vehicle productivity (figure 3.25) and vehicle utilization (figure 3.24) decreases gradually with the increase in C_2 parameter as expected.

Now, the question is, what exactly are the differences between the two disutility functions i.e., quadratic and linear, used ?

It can be easily observed that if two functions have same mean then quadratic function will assign more customers disutility to longer deviations than the linear function. Thus the difference between the two functions lie in the distribution of deviation in the schedules.

For $C_2 = 0.7$ and $C_1 = 5$ mean deviation is almost same (around 5 minutes). We have plotted a graph giving the information of deviation for these values of C_2 and C_1 (see figure 3.26). It can be easily observed from figure 3.26 that for linear disutility function ($C_1 = 5$) there are more customers having deviation higher than 20 minutes as compared to number of customers when quadratic disutility i assumed ($C_2 = 0.7$).

The next question arises that which form of disutility should be used in practice? This is a policy issue handled differently by different operators. But for large time windows the quadratic function may represent more realistic disutility function as it will generate schedules with preferable distribution of deviations.

(iii) Parameter C_3 :

This parameter represents the weight placed on the linear term of excess side time in the disutility function. In this part of investigation it is assumed that $DU_i^r = C_3\, y_i$. As far as deviation from desired time is concerned it hovers around 11 minutes as shown in figure 3.27. This can be easily understood as emphasis on excess ride time disutility does not result into improvement in the deviation from desired time.

Effect of varying parameter C3 on deviation from desired time (pickup or delivery)

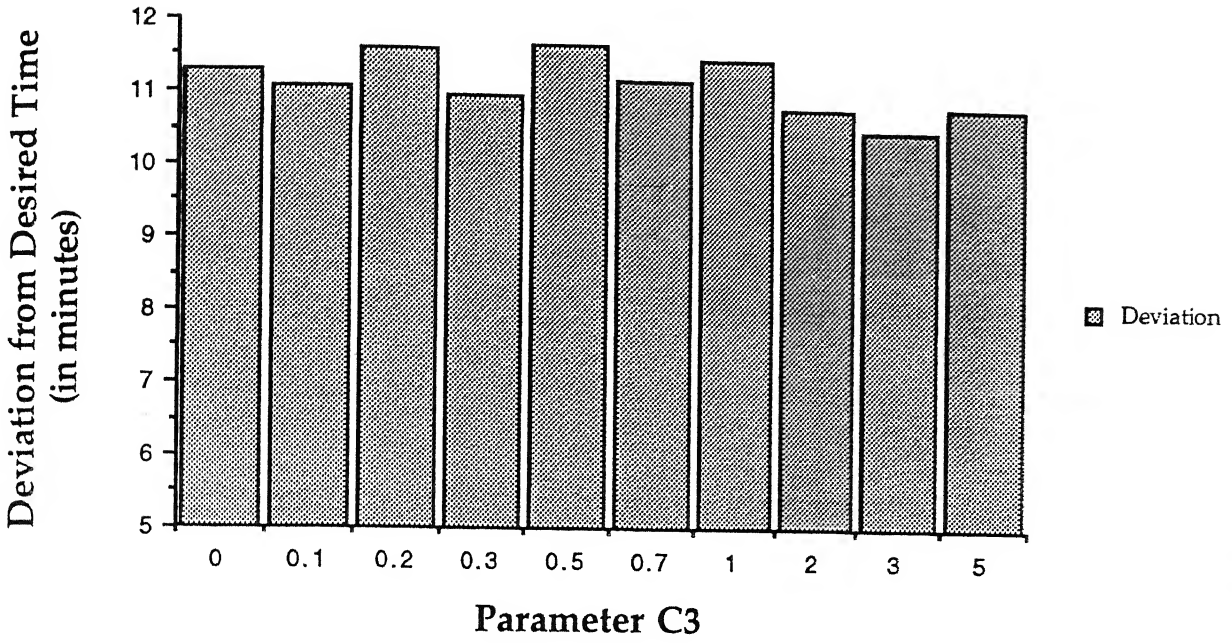


Figure 3.27

Effect of varying Parameter C3 on ride time ratio

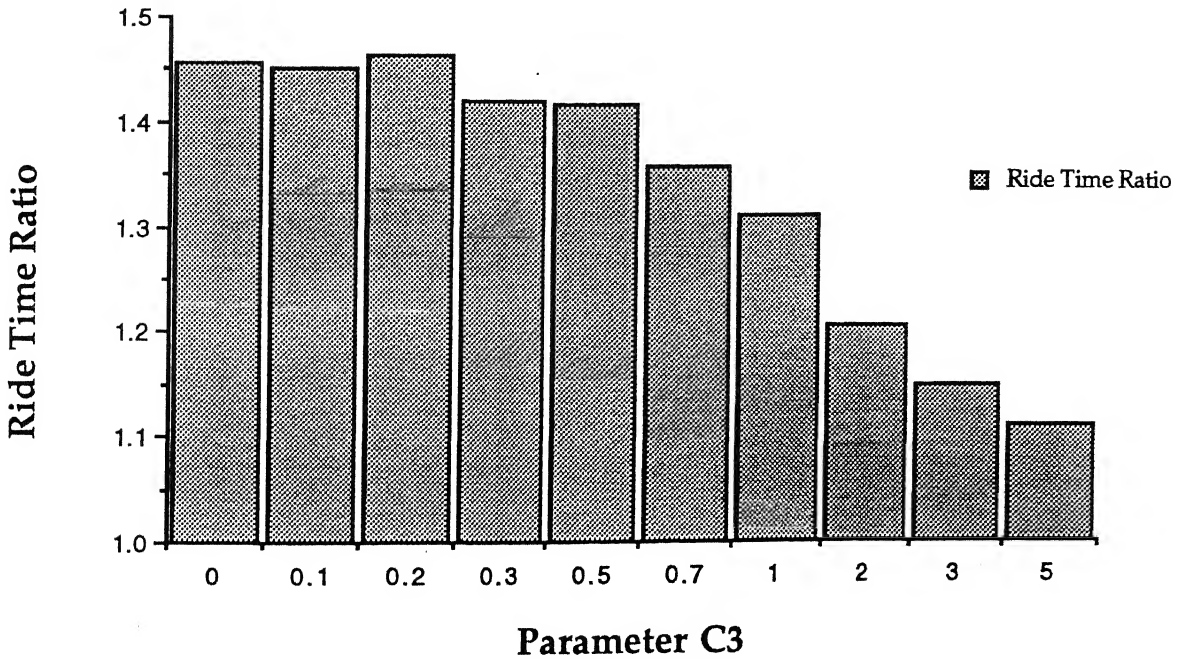


Figure 3.28

Effect of varying parameter C3 on active vehicle time and total vehicle time

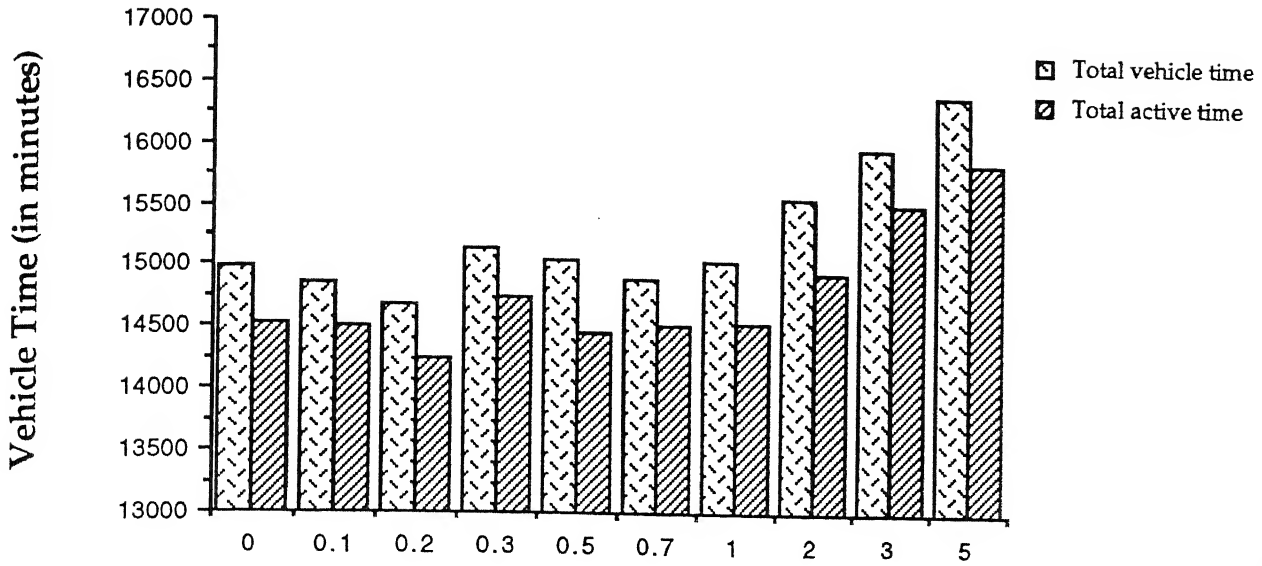


Figure 3.29

Effect of varying C3 on number of vehicles used

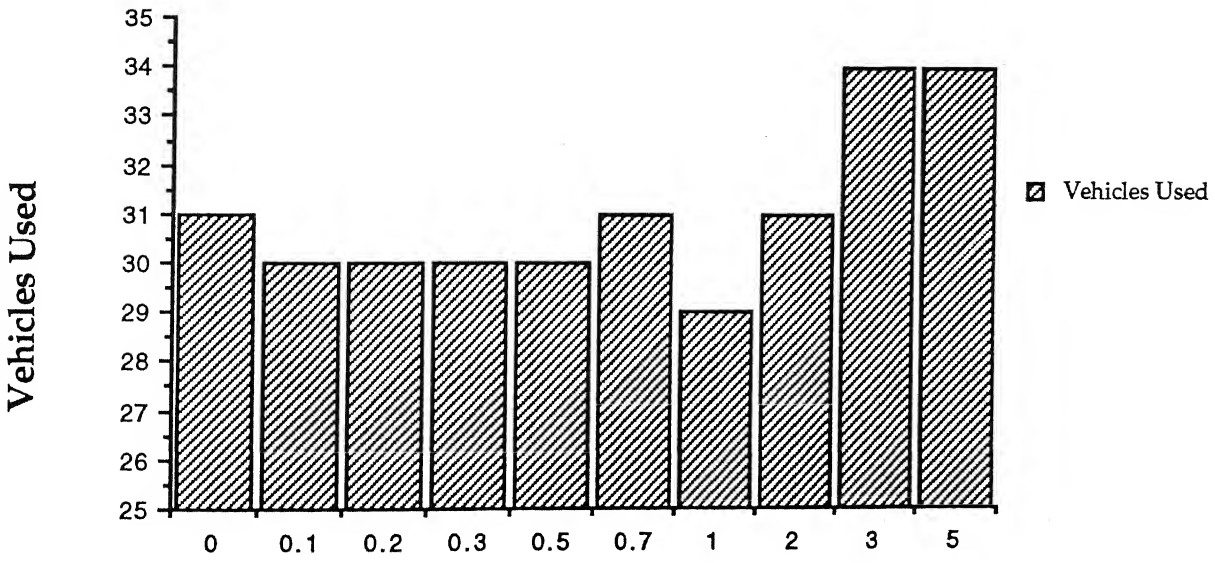


Figure 3.30

Effect of varying parameter C3 on vehicle productivity

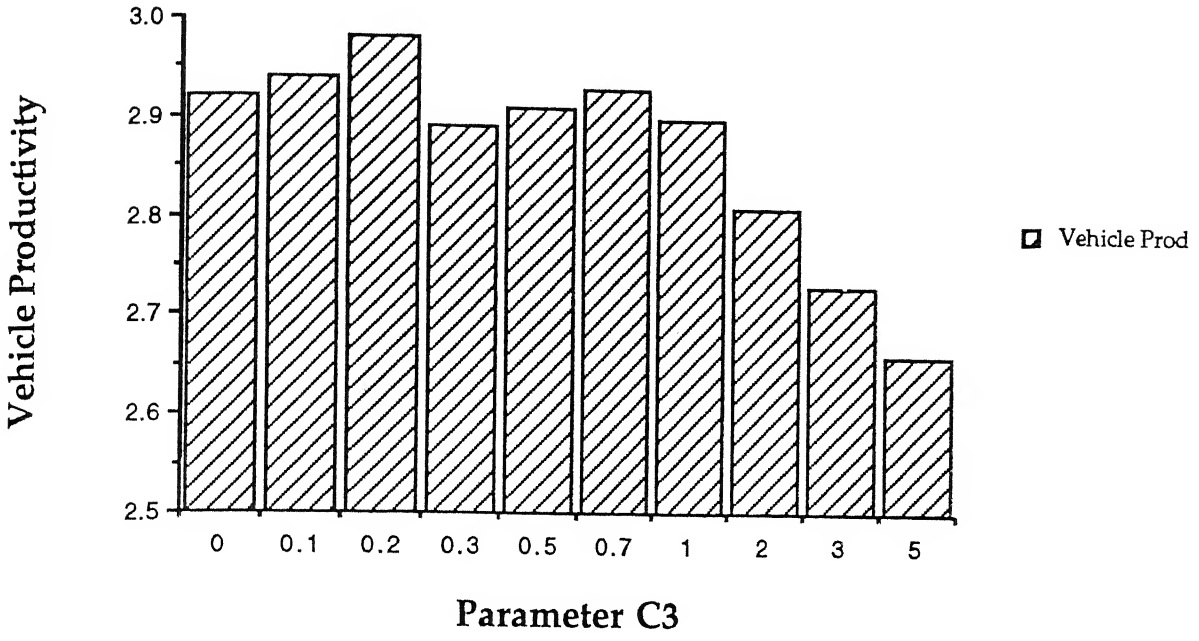


Figure 3.31

Effect of varying parameter C3 on vehicle utilization

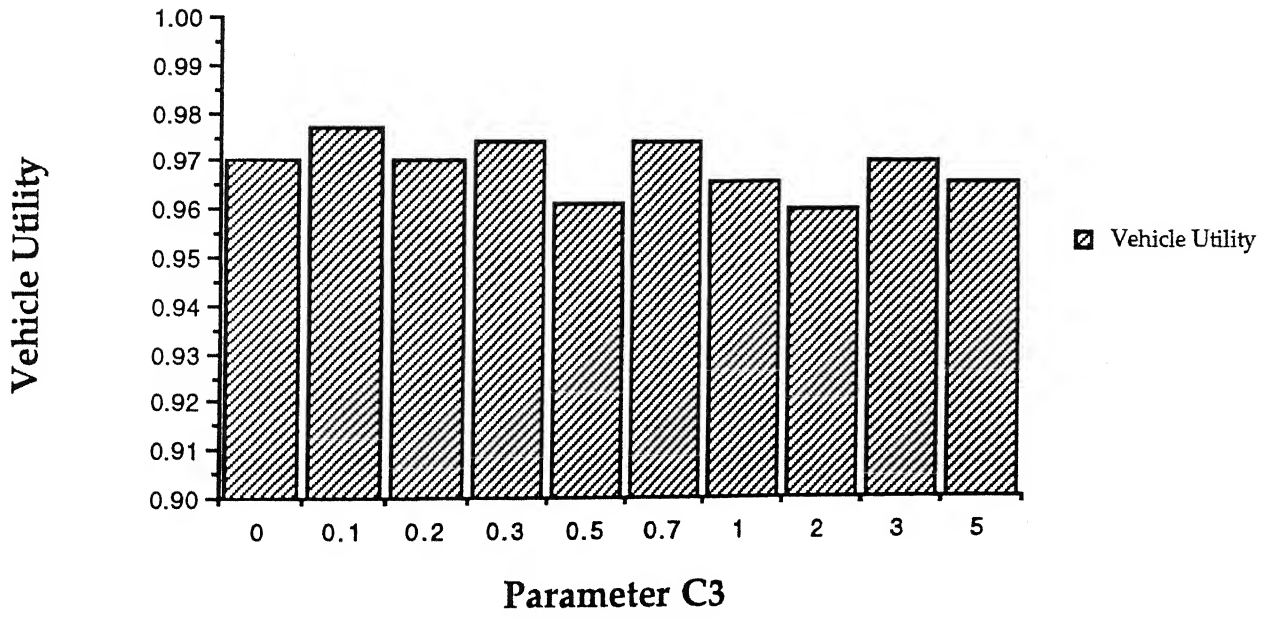


Figure 3.32

As depicted in the figure 3.28, ride time ratio for the base case i.e., with no customer disutility was 1.46. There is a 36% reduction in it from 1.46 to 1.1 for $C_3 = 5$.

The drastic reduction in Ride Time Ratio results in the increase of total vehicle time required, which shoots from base case value of 15000 minutes to 16500 minutes (figure 3.29).

Requirements in terms of number of vehicles required goes up from 31 to 34 (figure 3.30). Vehicle productivity reduces from 2.93 to 2.68 (figure 3.31).

Vehicle utilization remains almost constant at 96% (figure 3.32). This is because increase in the total vehicle time is followed by increase in the active vehicle time. Thus there is not much increase in the total slack as additional vehicle time is utilized in providing better service to the customers.

Here the important observation to make is that increasing the value of C_3 will not improve the things as far as deviation from desired time is concerned.

(iv) Parameter C_4 :

In this case quadratic excess ride time disutility function is assumed. Thus $DU_i^r = C_4 y_i^2$.

All the results as depicted in the graph are consistent with the observation made for the variation in parameter C_3. Effect on the deviation from desired pickup or delivery time by varying C_3 is absent as expected (figure 3.33). But ride time ratio decreases drastically and for value of 0.5 it even reduces below 1.1 (figure 3.34).

Vehicle time and number of vehicle required increases with the increase in C_4 (figure 3.35 and figure 3.36). Vehicle productivity reduces from 2.93 to

Effect of varying parameter C4 on deviation from desired specified time (pickup or delivery)

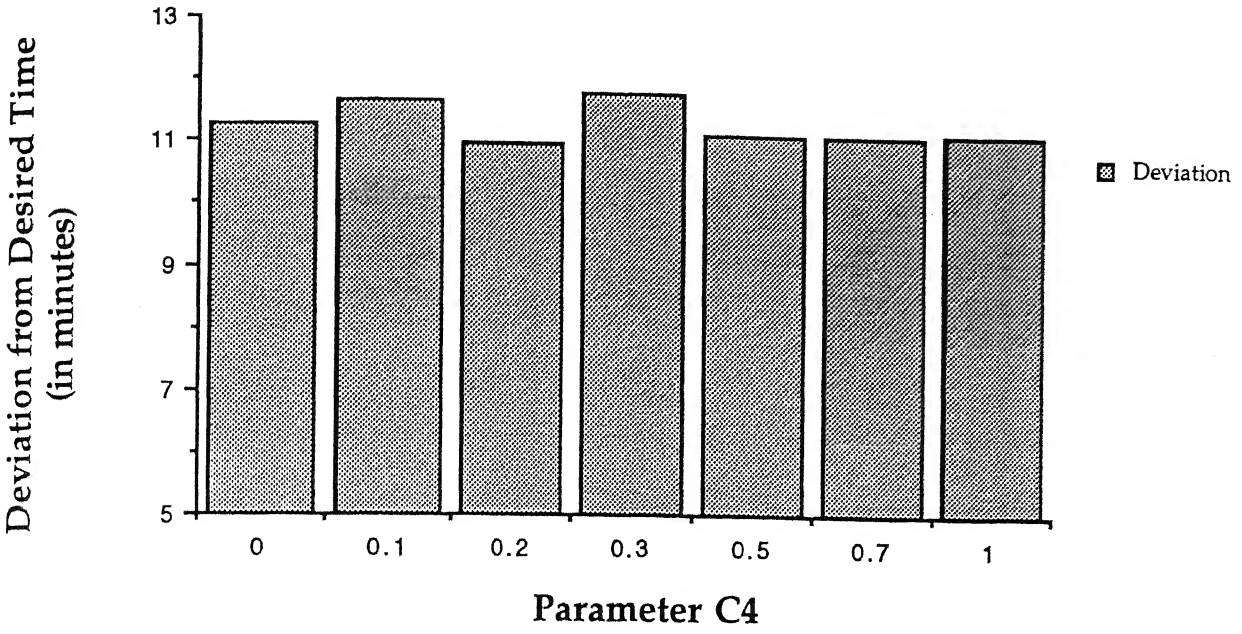


Figure 3.33

Effect of varying parameter C4 on ride time ratio

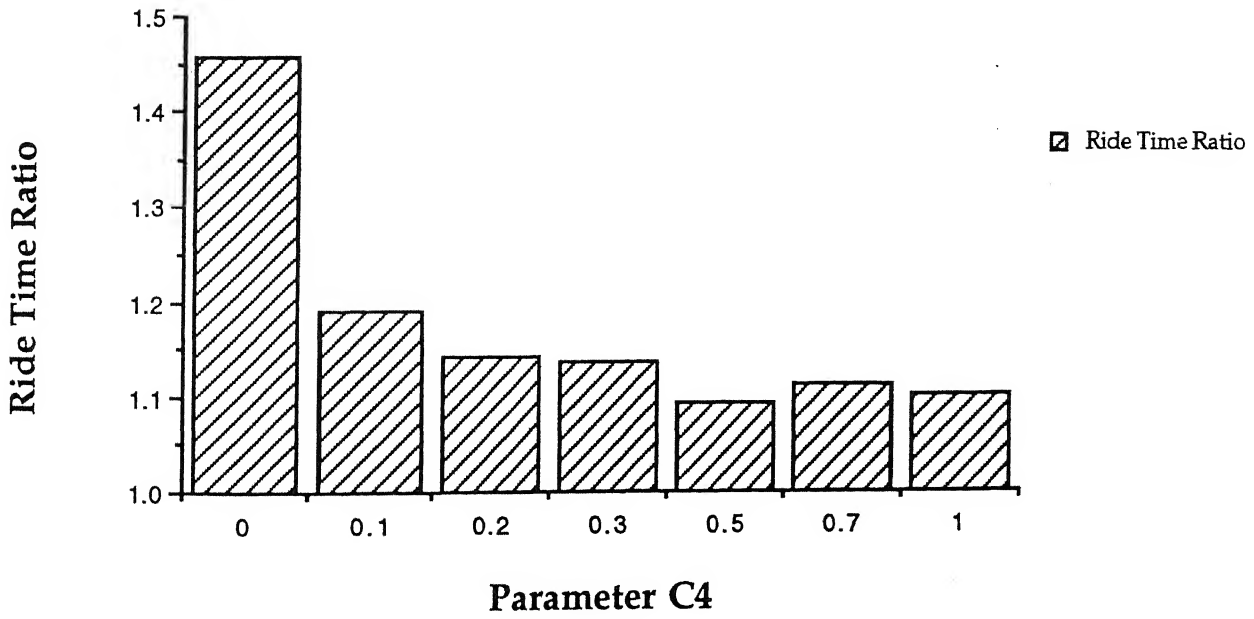


Figure 3.34

Effect of varying parameter C4 on active vehicle time and total vehicle time

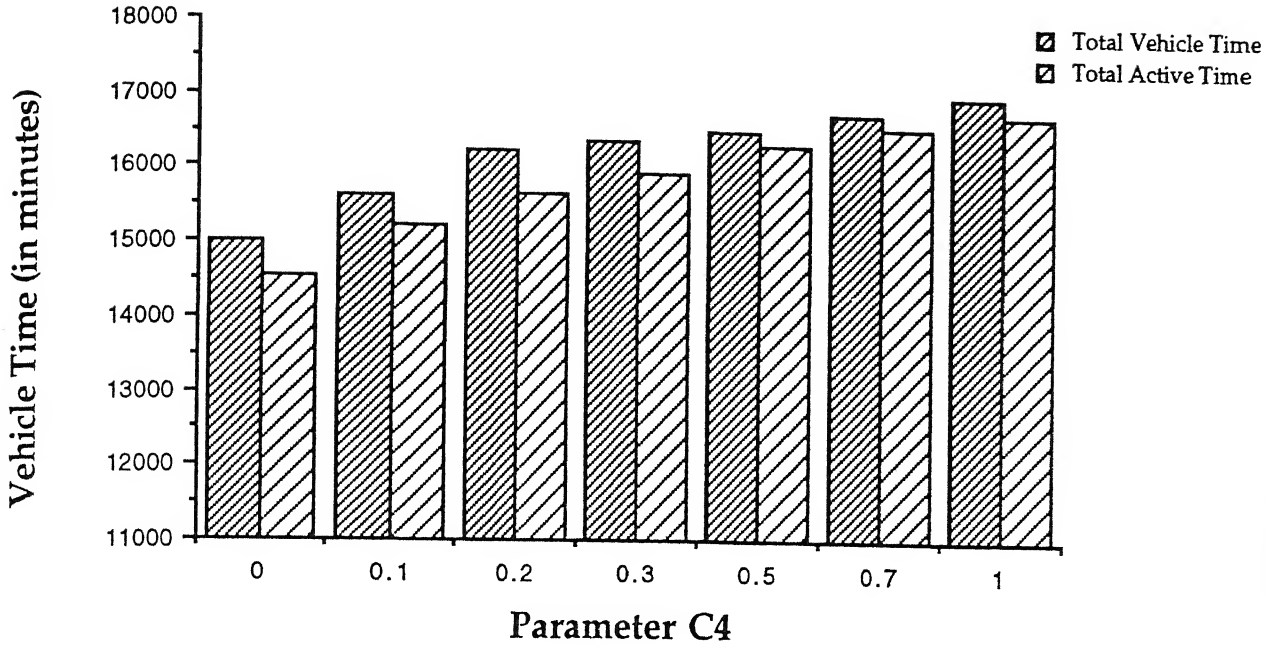


Figure 3.35

Effect of varying C4 on number of vehicles used

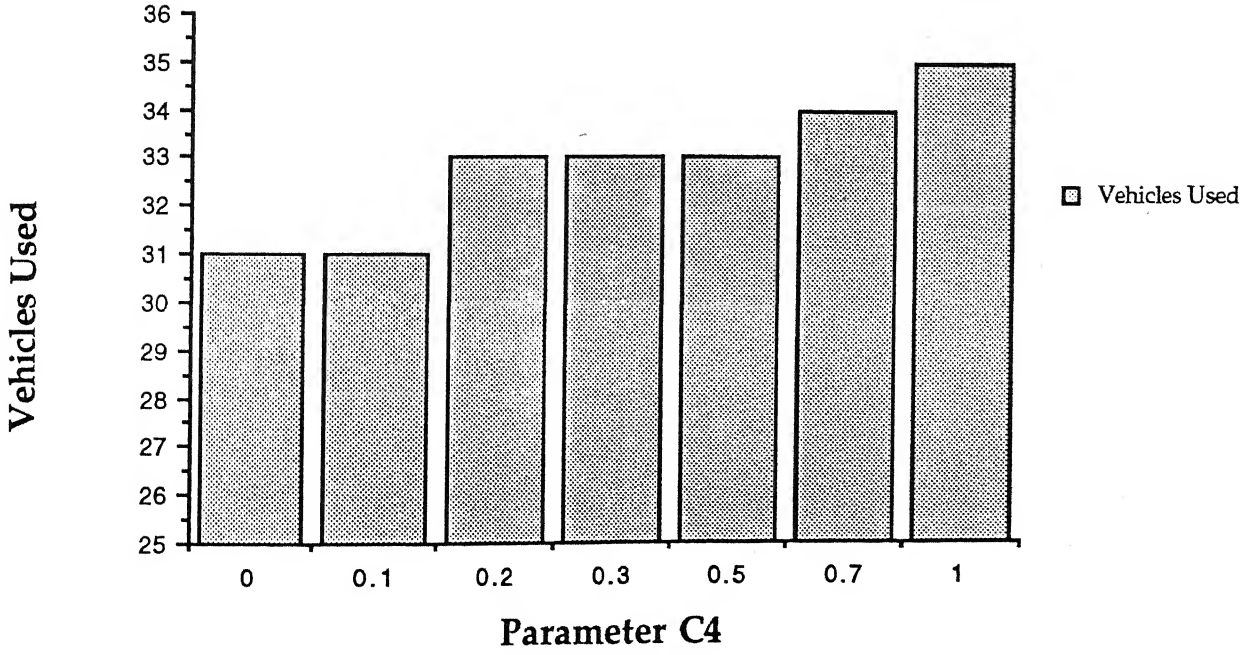


Figure 3.36

Effect of parameter C4 on vehicle productivity

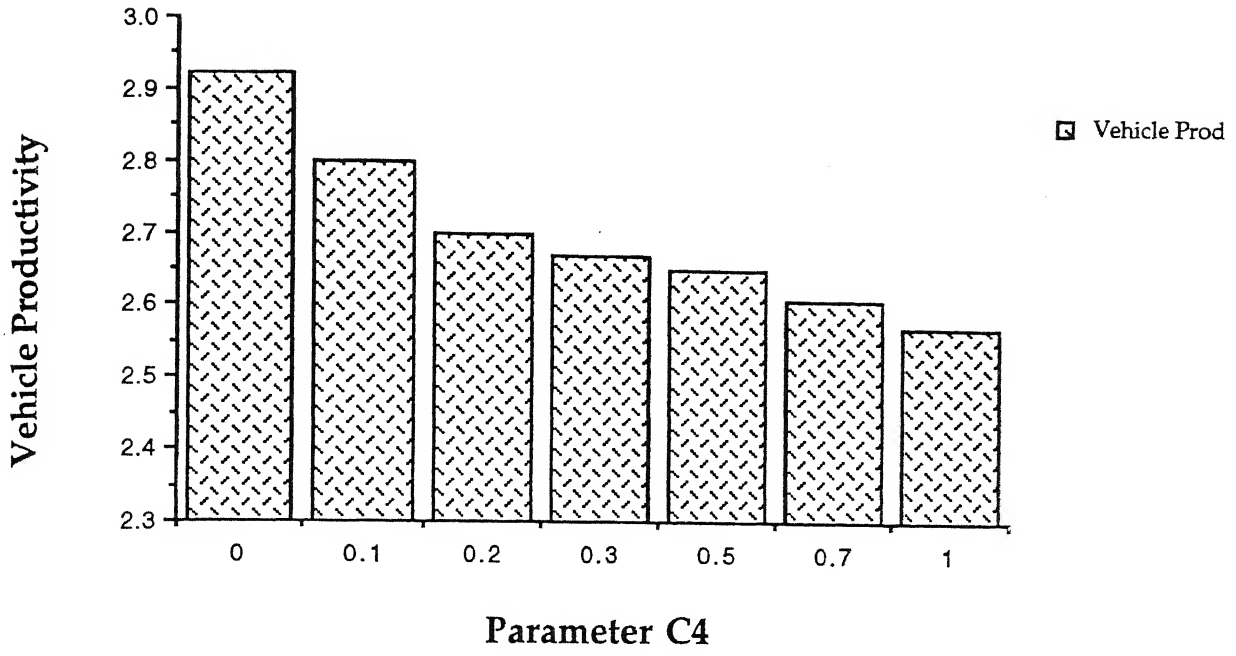


Figure 3.37

Effect of varying parameter C4 on vehicle utility

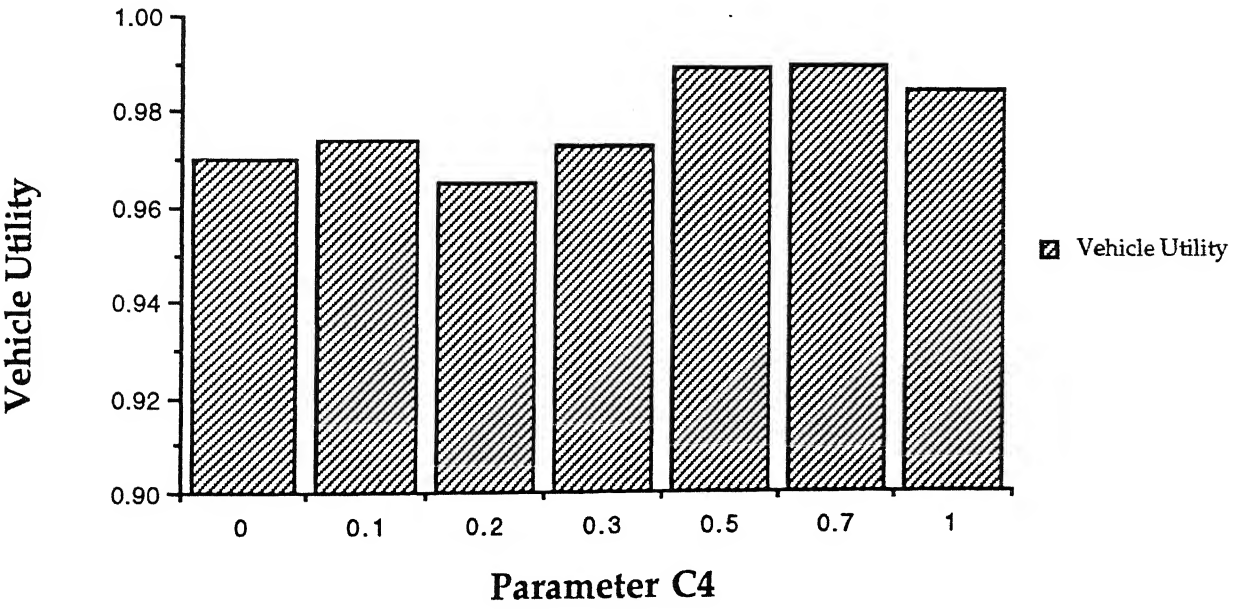


Figure 3.38

below 2.6 from base case to the case when C_4 is 1(figure 3.37). As shown in the figure 3.38 vehicle utility remains at 97%.

(v) Combined effect of C_1 and C_3 :

The disutility function is assumed to be a linear function of deviation from desired time and excess ride time. For our further experiment purpose, we will concentrate on what mix of C_1, C_3, C_5, C_6, C_7, C_8 will provide reasonable quality of solutions and will not consider parameters C_2 and C_4.

We have performed the experiment runs for combination of C_1 and C_3 such that C_1 varies from 1 to 3 and C_3 varies from 0.1 to 0.5.

We have intentionally chosen higher values of C_1 and lower values of C_3 because it will generate the schedules which will place more emphasis on the deviation from desired pickup or delivery time than the excess ride time. In general, we have assumed that less waiting time will be more preferred by the customers than more excess ride time.

Also as we have observed earlier that for variation in the values of C_3 would not affect deviation from desired time and it usually hovers around 11 minutes. Thus we need to place emphasis on the parameter C_1 to generate the schedules having low deviation. Also placing emphasis on C_1 results in the reduction of excess ride time. Thus we have done experiment with such combination of C_1 and C_3 in which $C_1 > C_3$.

For $C_1 = 1$ the deviation from desired time is approximately 7.5 minutes for all the values of C_3, which is high compared to deviation corresponding to higher values of C_1 (figure 3.39)

From figure 3.40 it can be easily observed that the ride time ratio, for $C_1 = 2$ and $C_1 = 3$ with C_3 varies from 0.1 to 0.5, hangs around 1.3. It is not very sensitive to the variation in C_3. There is 11% variation in the ride time ratio

Effect of varying parameter C1 and C3 on deviation from desired service time (pickup or delivery)

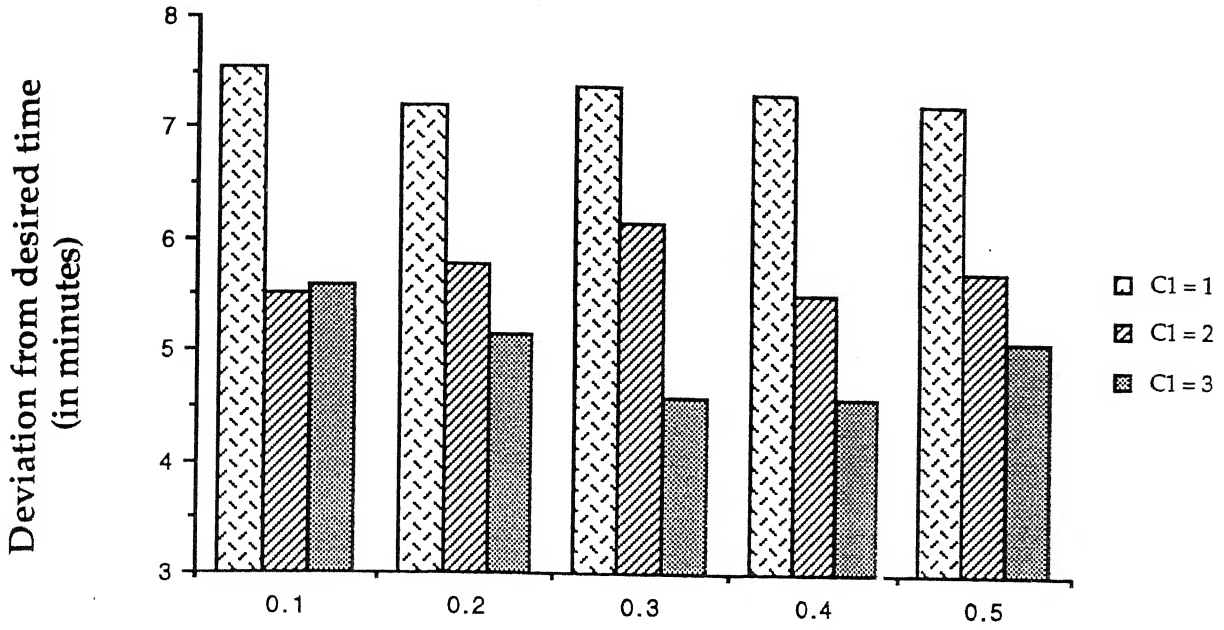


Figure 3.39

Effect of C1 and C3 on ride time ratio

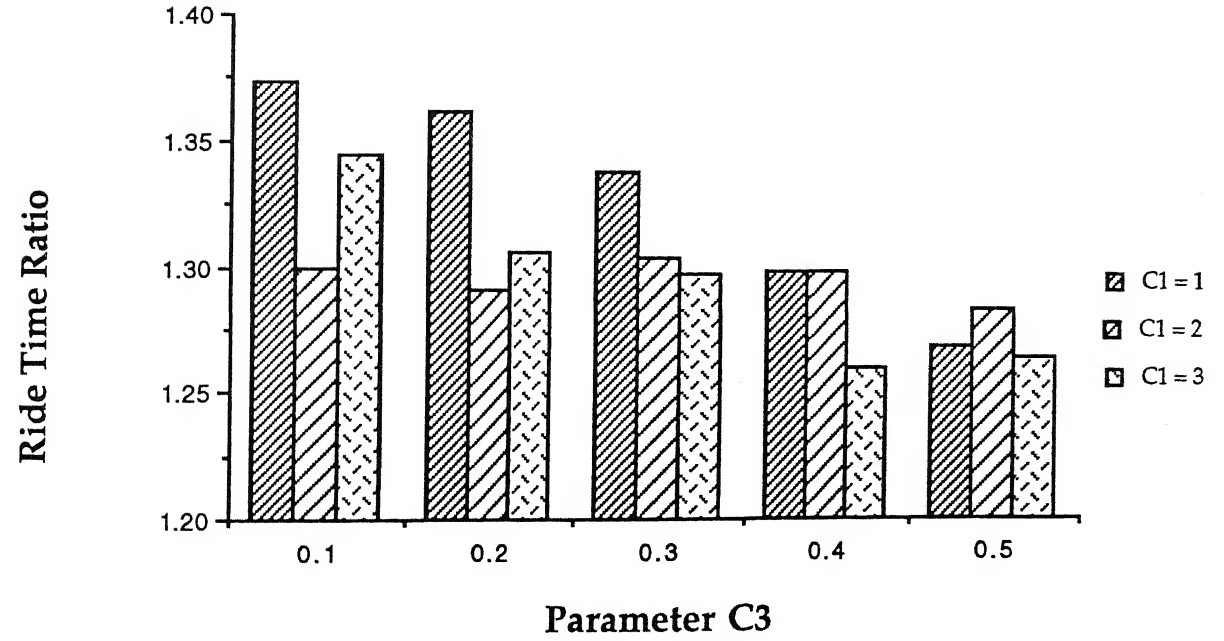


Figure 3.40

Effect of C1 and C3 on number of vehicles used

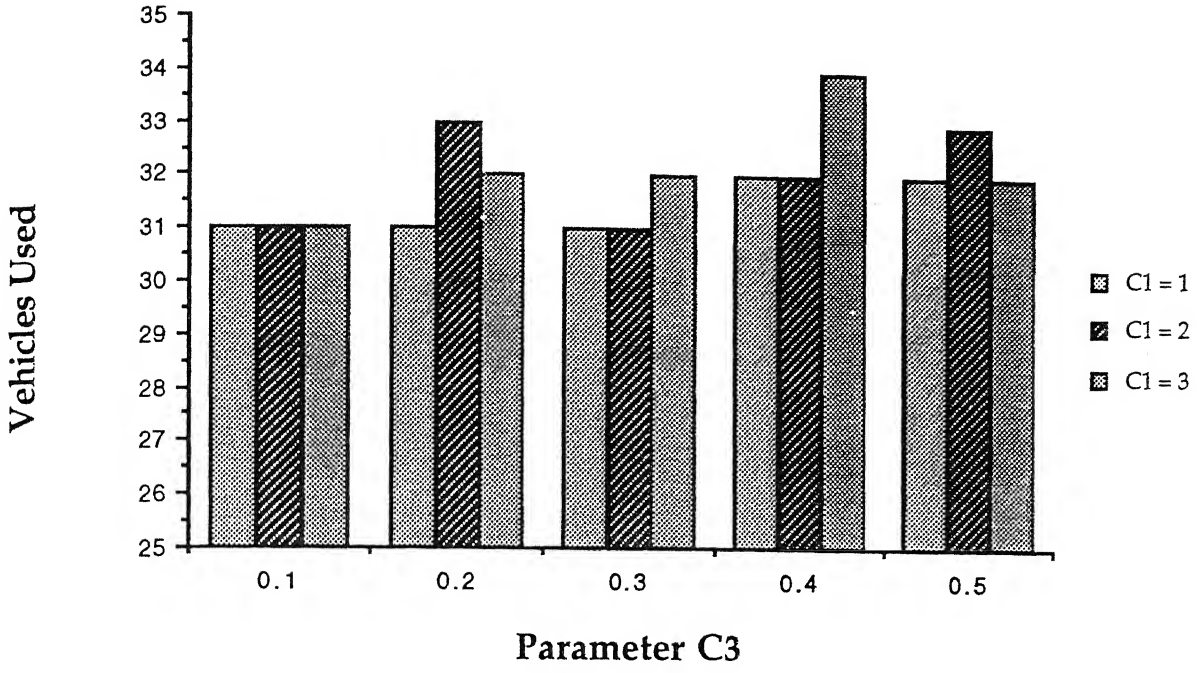


Figure 3.41

Effect of C1 and C3 on active time required by vehicles

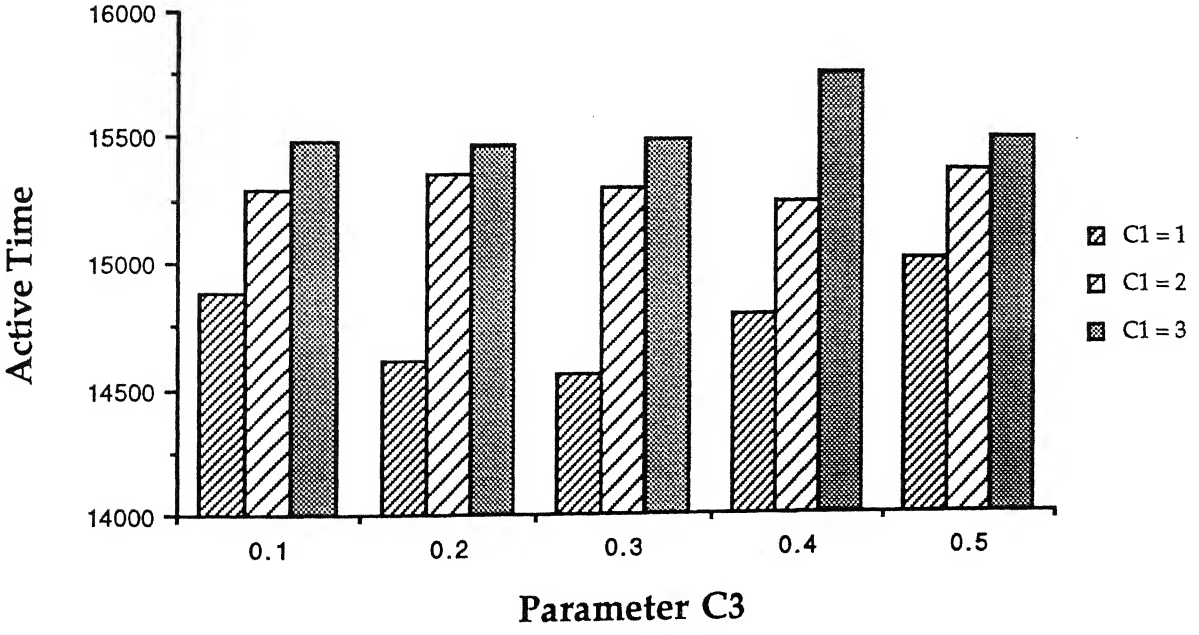


Figure 3.42

for $C_1 = 1$ and then varying as in this case emphasis on both type of disutility is comparable in objective function.

For $C_3 = 0.1$ vehicle required are minimum (figure 3.41).

It can be observed from figure 3.42 that when $C_1 = 1$, increase in active vehicle time required is more sensitive to variation in C_3. For higher values of C_1, increase in active vehicle time required is not sensitive to variation in C_3.

From above observation, we think that keeping $C_1 = 3$ would ensure less deviation and with this value of C_1 we can select smaller value of C_3 as higher value may not be that useful in reduction of excess ride time because at $C_1 = 3$ ride time ratio is less sensitive to the variation in C_3. Thus we think that $C_1 = 3$ and $C_3 = 0.1$ will make a good proposition as this will result in smaller deviation time and reasonable ride time ratio at reasonable vehicle resources.

From now onwards we will use these values for our further experimentation purpose.

(vi) Parameter C_5 and C_6 :

Parameters C_5 and C_6 in the vehicle resource function provide with an alternative vehicle resource function which does not employ U_i. All runs so far have used U_i in the vehicle resource function by specifying $C_5 = 0$, $C_6 = 0$ and $C_7 \neq 0$, $C_8 \neq 0$. That is

$$VC_i = U_i (C_7 z_i + C_8 w_i) \qquad (3.35)$$

If instead $C_7 = 0$, $C_8 = 0$ and $C_5 \neq 0$, $C_6 \neq 0$, we have the alternative function :

$$VC_i = C_5 z_i + C_6 w_i \qquad (3.36)$$

As explained in section 3.6, U_i is used to conserve vehicle resources during high demand periods and to put relatively more emphasis on the quality of service during periods of low demand. Through experiment we have tried to analyze that how exactly this is taking place.

For this purpose we have chosen five different instances of 500 requests. The number of persons associated with each of these instances are plotted on the x-axis of the figures from 3.43 to 3.46. For each instance we have gathered various statistics for two different runs. In first run, we have used term given in equation 3.36 in the objective function to represent system operating cost, and in second run we have used term given in equation 3.35 in the objective function. We for our purpose used the values of different parameters as C_1 = 3, C_2 = 0, C_3 = 0.1and C_4 = 0. For first run C_5 = 0, C_6 = 0, C_7 = 1.2 and C_8 = 0.7 is used i.e., we considered the effect of U_i. For second we didn't considered the effect of U_i by taking C_5 = 1.2, C_6 = 0.7, C_7 = 0 and C_8 = 1.2.

As it can be observed from the figure 3.43, that except in last instance, number of vehicles required reduces when we have used U_i in the objective function. Similarly, in figure 3.44 active time required by the vehicles reduces when U_i is used in the objective function. This means that in the high demand using U_i conserves vehicle resources.

The cost of saving vehicle resources is paid by the quality of service provided. As shown in the figure 3.45 deviation from desired time increases by almost 1.5 minutes on an average. Also there is increase in the excess ride time traveled by the customers (figure 3.46).

(vii) Effect of varying the MRT function :

The MRT (maximum ride time) function used by us is

MRT = A + B*DRT.

Effect of using U in objetive function on number of vehicles used

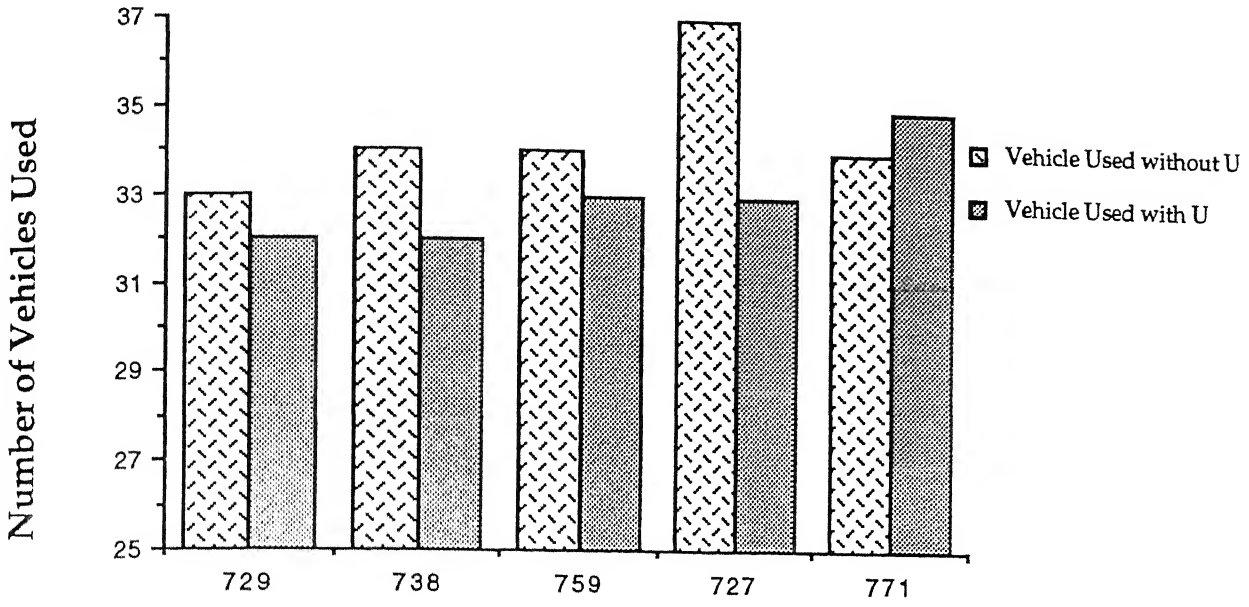


Figure 3.43

Effect of using U in objective function on total vehicle time and active vehicle time

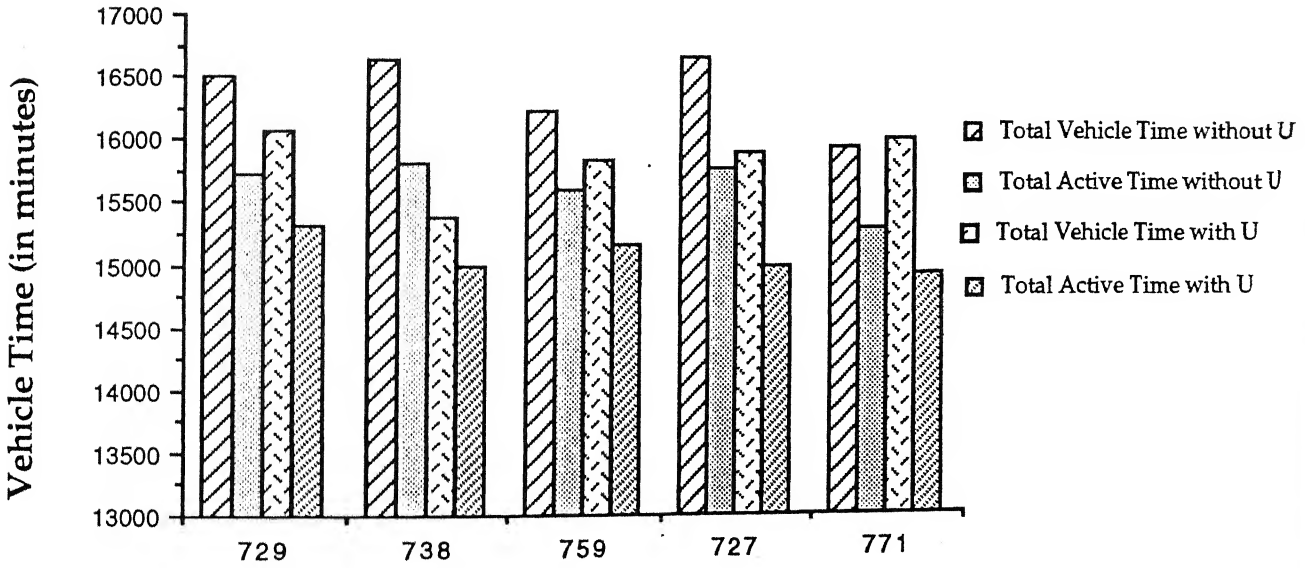


Figure 3.44

Effect of using U in objetive function on deviation from desired service time (pickup or delivery)

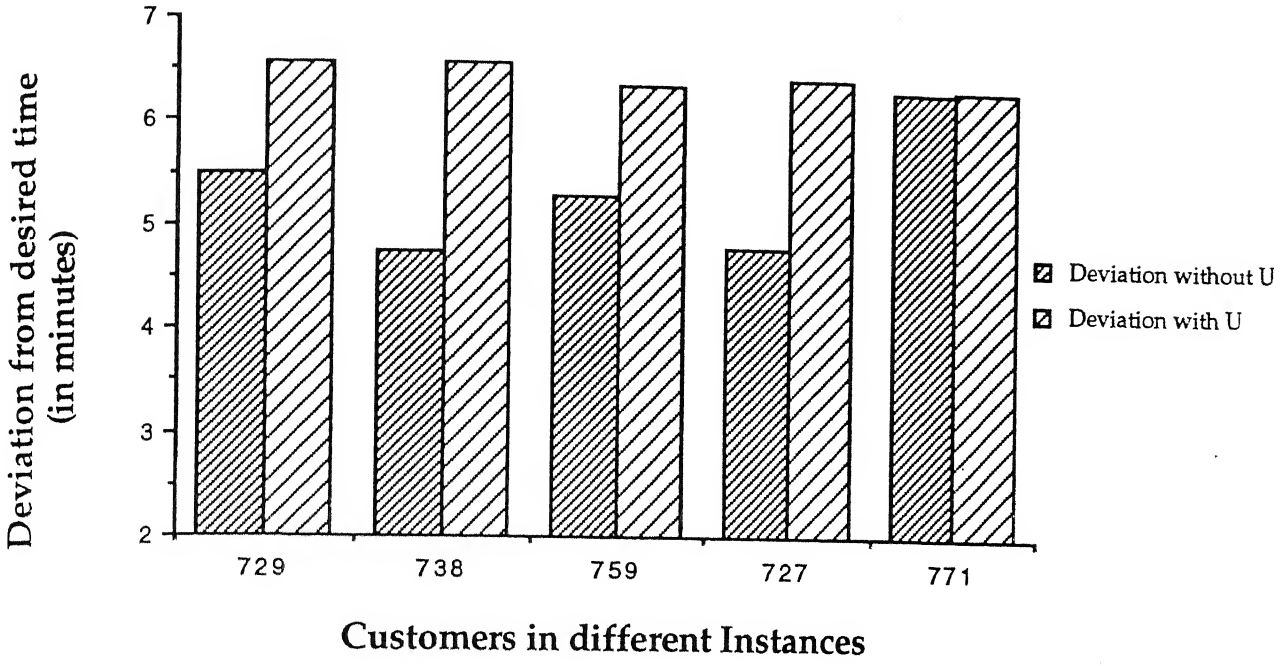


Figure 3.45

Effect of using U in objetive function on ride time ratio

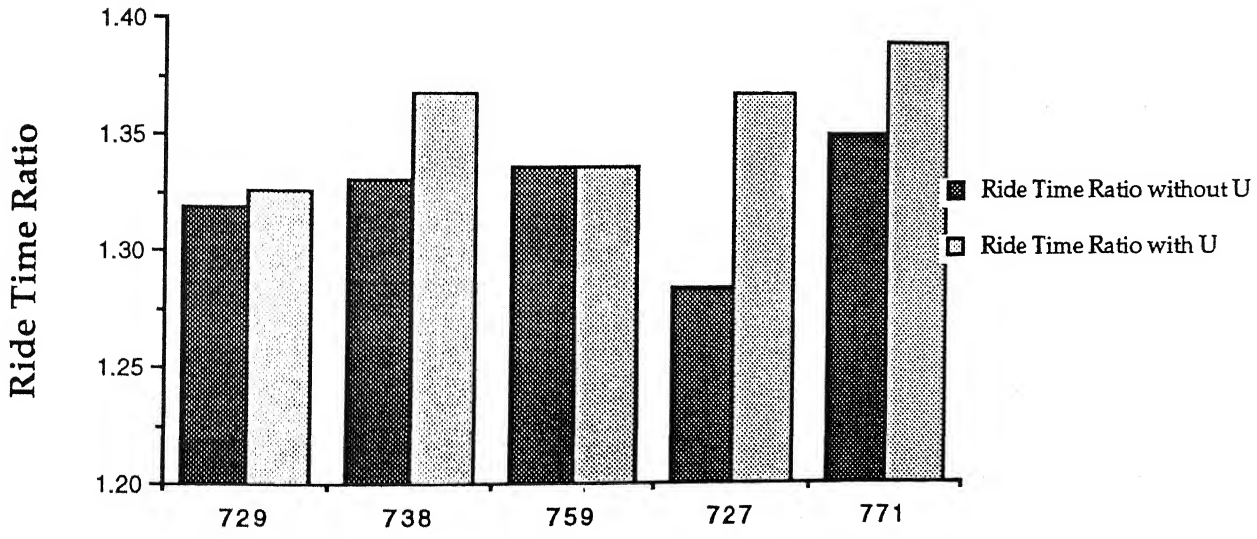


Figure 3.46

where A = 10 minutes, B = 1.5 for expensive service and 1.7 for normal service, DRT is the direct ride time between two nodes.

For our experiment purpose we have chosen $C_1 = 0$, $C_2 = 0$, $C_4 = 0$, $C_5 = 0$, $C_6 = 0$, $C_7 = 1.2$ and $C_8 = 0.7$. We have varied values of C_3 from 0 to 2. Also we have assumed same MRT function for both type of quality service.

From figure 3.47 it can be observed that as we relax the MRT constraint ride time ratio increases. This increase is more evident for the lower values of C_3, because of lesser penalty is associated with it in the objective function. This means at this time main emphasis is to conserve the vehicle resources. Change in B will change the constraint on MRT which is directly reflected with increase in B. For higher values of C_3, for example 2, there is very less variation in excess ride time even for value of B = 2. Because in this case the emphasis is on reducing disutility due to excess ride time. From figure 3.48, it can be seen that there is no impact of variation of B on deviation from desired time and it remains generally around 12 minutes.

It can be easily observed from figure 3.49 that for lower values of C_3, as the value of B is increased, the number of vehicles required reduces. But, again, for higher values of C_3 number of vehicles required remains constant. This is because of the fact that as we give higher emphasis to the excess ride time disutility, vehicles are added to the system very quickly even for B = 2. From figure 3.50 again the similar trends are evident. Total vehicle time and active vehicle time are more sensitive for lower values of C_3.

From figure 3.51 it can be easily observed that vehicle utilization increases with the relaxing of the MRT constraint. Also, from figure 3.52, vehicle productivity increases for higher values of B. But when C_3 is increased to 2, vehicle productivity becomes constant. This is again because of the fact that

Effect of varying B in the equation
MRT = A + B*DRT on ride time ratio

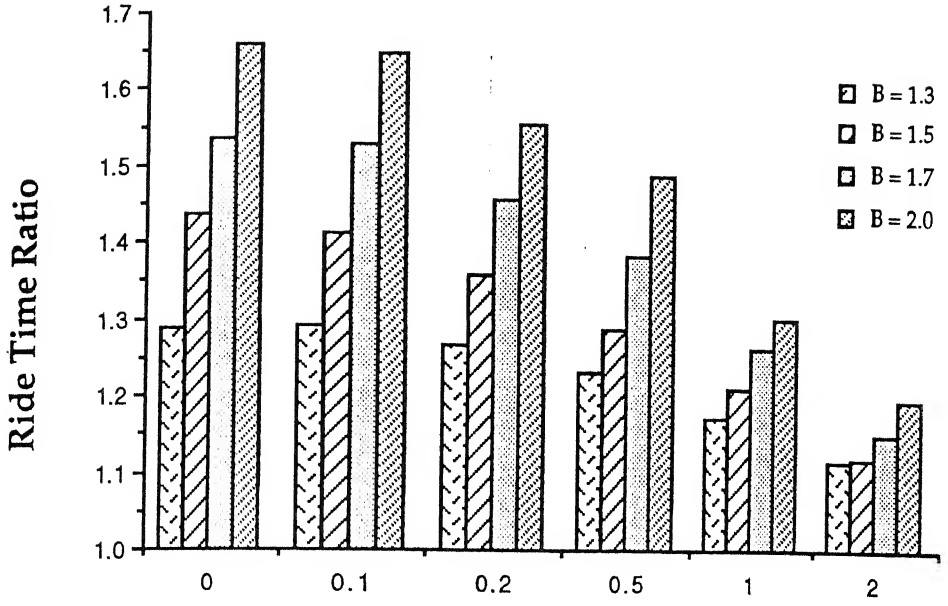


Figure 3.47

Effect of varying B in the equation MRT = A + B*DRT
on deviation from desired pickup or delivery time

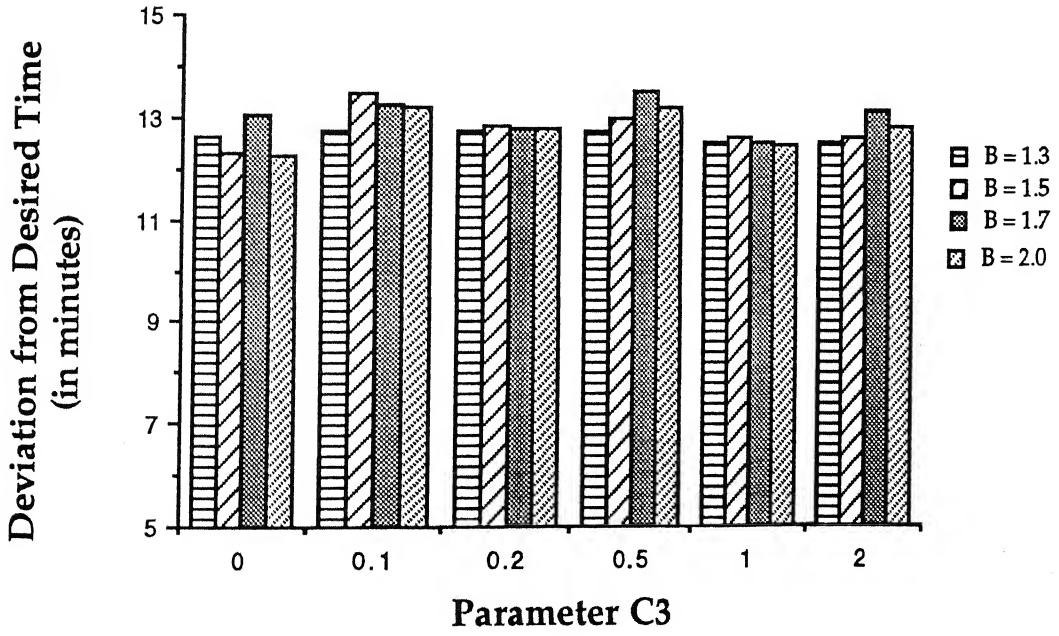


Figure 3.48

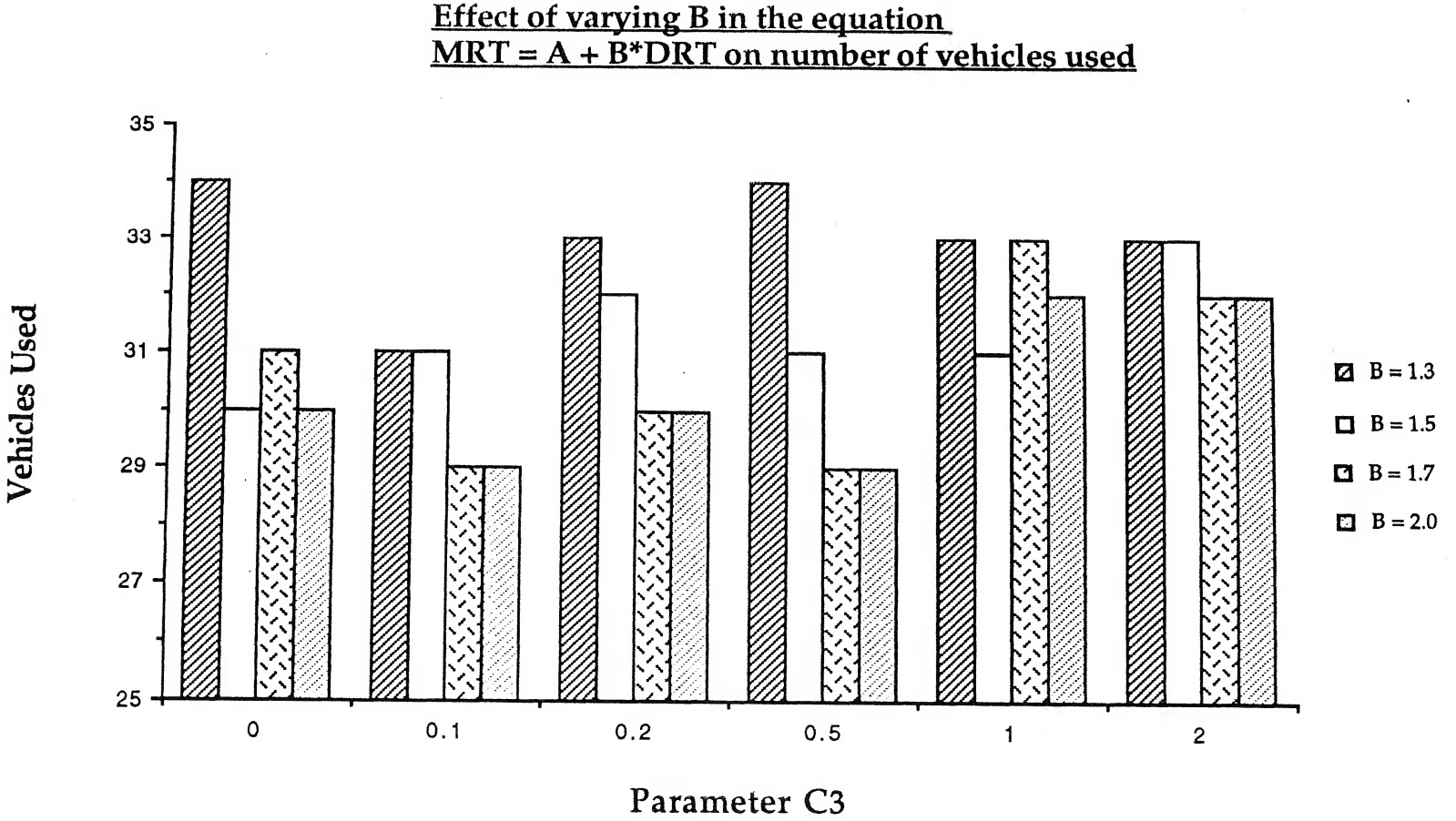

Effect of varying B in the equation
MRT = A + B*DRT on number of vehicles used
Vehicles Used
35
33
31
29
27
25
0
0.1
0.2
0.5
1
2
Parameter C3
B = 1.3
B = 1.5
B = 1.7
B = 2.0

Figure 3.49

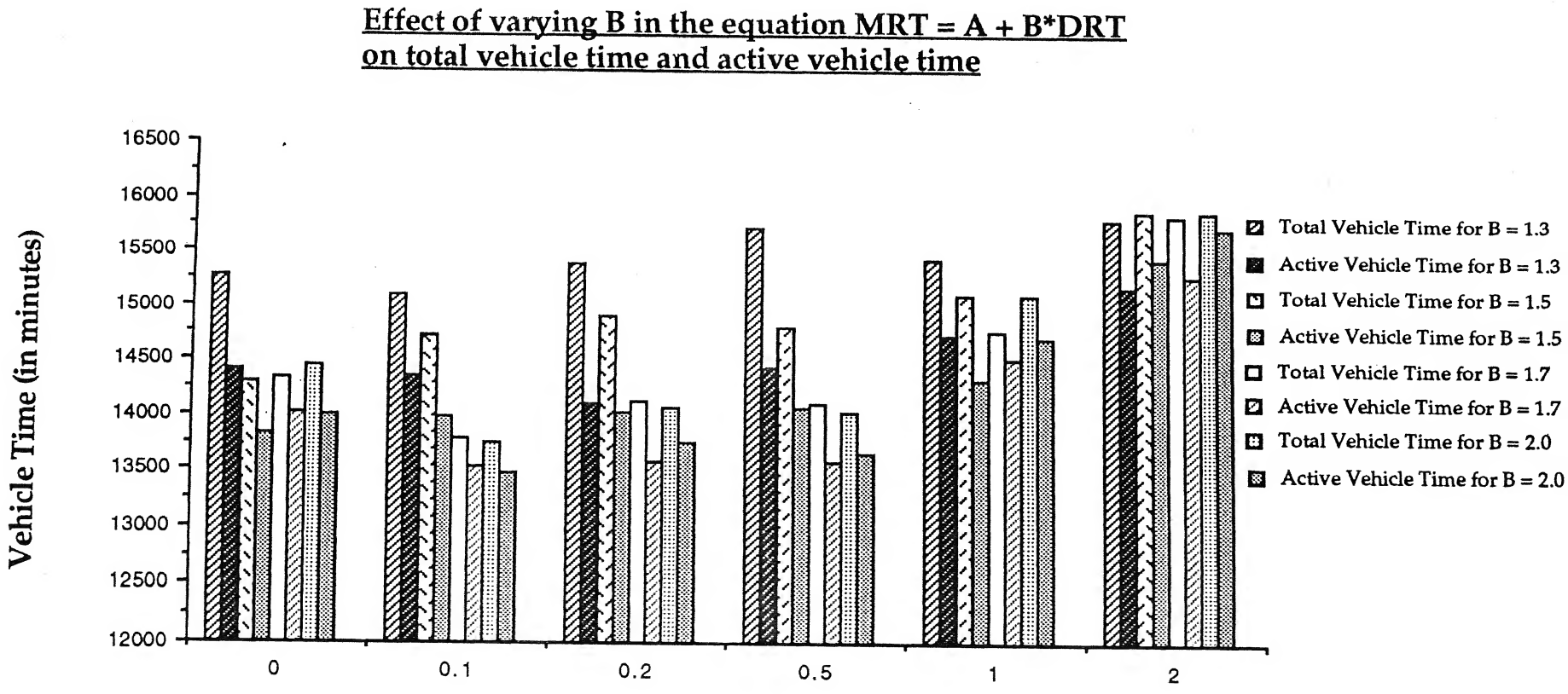

Effect of varying B in the equation MRT = A + B*DRT
on total vehicle time and active vehicle time
Vehicle Time (in minutes)
16500
16000
15500
15000
14500
14000
13500
13000
12500
12000
0
0.1
0.2
0.5
1
2
Parameter C3
Total Vehicle Time for B = 1.3
Active Vehicle Time for B = 1.3
Total Vehicle Time for B = 1.5
Active Vehicle Time for B = 1.5
Total Vehicle Time for B = 1.7
Active Vehicle Time for B = 1.7
Total Vehicle Time for B = 2.0
Active Vehicle Time for B = 2.0

Figure 3.50

Effect of varying B in the equation MRT = A + B*DRT on vehicle productivity

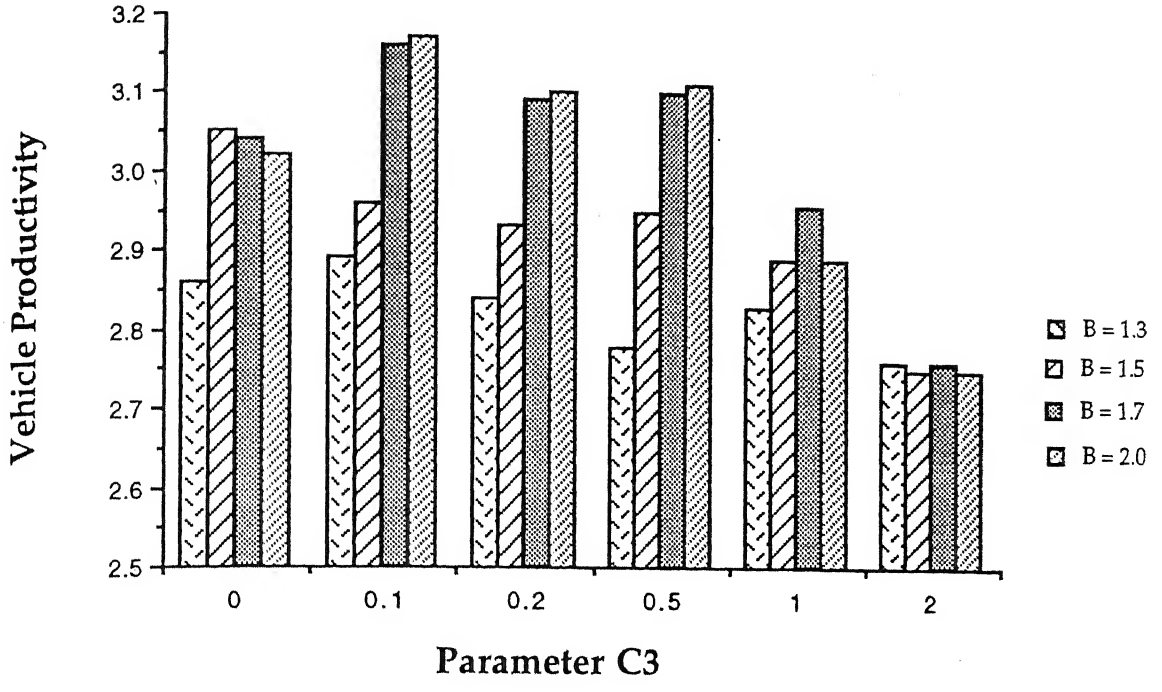
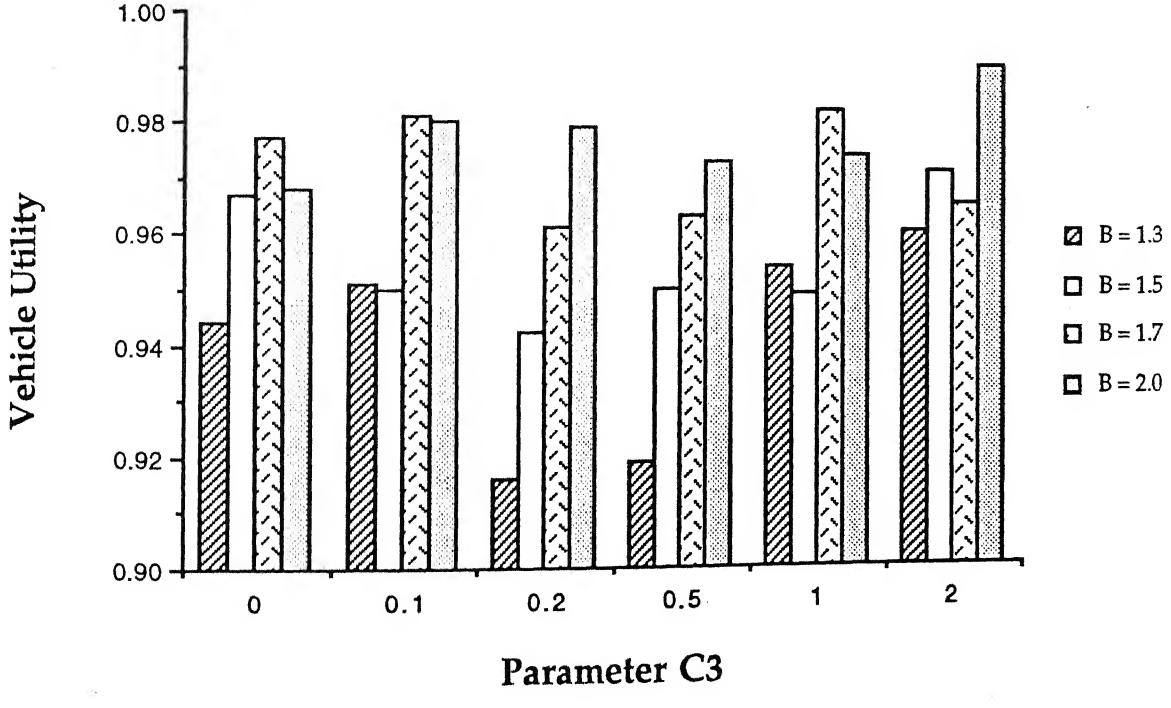


Figure 3.52

Effect of varying B in the equation MRT = A + B*DRT on vehicle utilization



for lower values of C_3 more emphasis is on the conservation of vehicle resources. Thus increasing the B will result in higher vehicle productivity.

3.7.5 Investigations of the effects of different Pool filling strategies

For the second part of our investigation using simulated data, we focus on the effects of considering multiple candidates for insertions. Both Pool refilling strategies are tested.

(a) Immediate Refill Strategy vs. Periodic Refill Strategy

We have run our program for 5 different instances each consisting of 500 customer requests. We then have plotted the average value of objective function in the graph. As it can be seen that we did not found any improvement in case of Immediate Refill for higher pool sizes. But for Periodic Refill Strategy, we found improvements for the higher value of pool size (figure 3.53 and figure 3.54).

Now the question arises that why did we find the improvement for periodic refill?

The difference between the two strategy is that in case of periodic refill the order of insertion of customers are almost their EPT order. This is because of the fact that next set of least EPT customers comes in the pool for consideration only when previous set has already been assigned.

But in the case of Immediate Refill there exists possibility of insertions which are not in the EPT order.

We have plotted graph for immediate refill strategy and periodic refill strategy depicting the incremental cost of insertion versus the order of customer's insertion and EPT of customers versus order of customer's insertion.

Effect of pool size on objective function for assigning advance request using immediate pool refilling

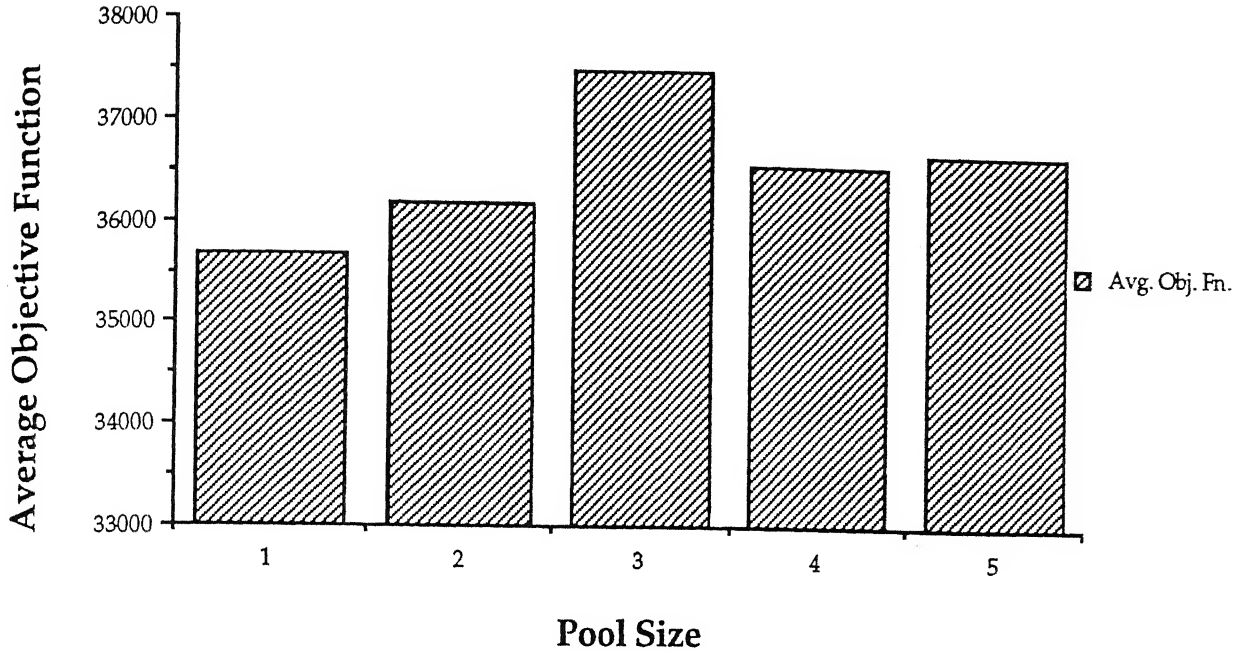
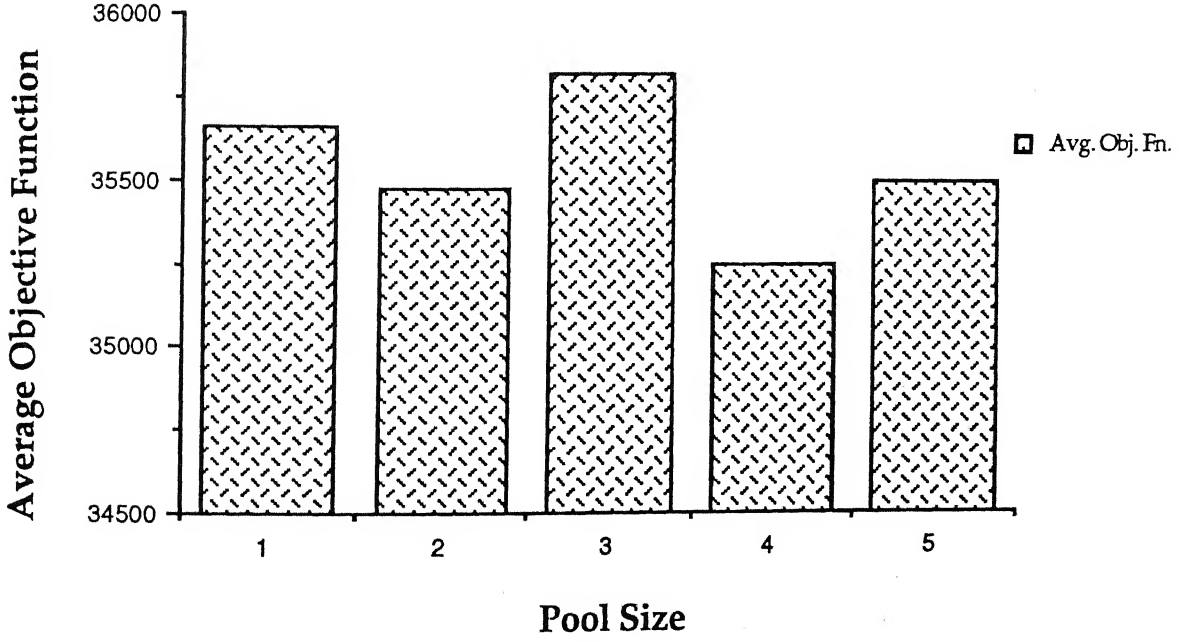


Figure 3.53

Effect of pool size on objective function for assigning advance request using periodic pool refilling



As it can be observed in the case of immediate refill that there exists sudden rise in the incremental cost of insertion (figure 3.55) and corresponding to these, as shown in figure 3.56 that, there exists an insertion which is "out of EPT order".

Whereas in case of periodic refill there is no "out of order EPT" insertion (figure 3.57 and figure 3.58).

The computation effort required for considering simultaneously more than one customer at a time as candidates for the next insertion is illustrated in Figure 3.59. Both pool-refilling strategies were investigated over a range of pool sizes. The positive correlation between computation time and pool size, a depicted in Figure is well expected since more candidates in the pool would certainly require more computations before ADARTW can choose the best candidate. However, such increasing trends are more significant for the immediate-refill strategy than for the periodic-refill strategy. The differences in computation time between the two strategies are mainly due to the updating efforts involved after a customer is chosen to leave the pool. To demonstrate this point, let us assume that the pool size is d and there are N customers on the subscription list. When a customer in the candidate pool is inserted into the work-schedule of vehicle j, it is required that for each of the remaining candidates in the pool their insertion costs to vehicle j be updated. We use the term "one update" to denote the evaluation of the insertion cot of a customer to a vehicle. Under the immediate-refill strategy, the number of updates required when a customer leaves the pool is d-1 since there are always d-1 unassigned customers in the pool. Consequently, the updating procedure would be executed (d-1)N times in total sine there are N customers in the list. Under the periodic-refill strategy, the number of updates required when a customer leaves the pool is not constant. It varies between 0 and d-1 depending on the number of unassigned customers in the pool at the time of

Growth of objective function as the order of customer's insertion for immediate refilling of pool

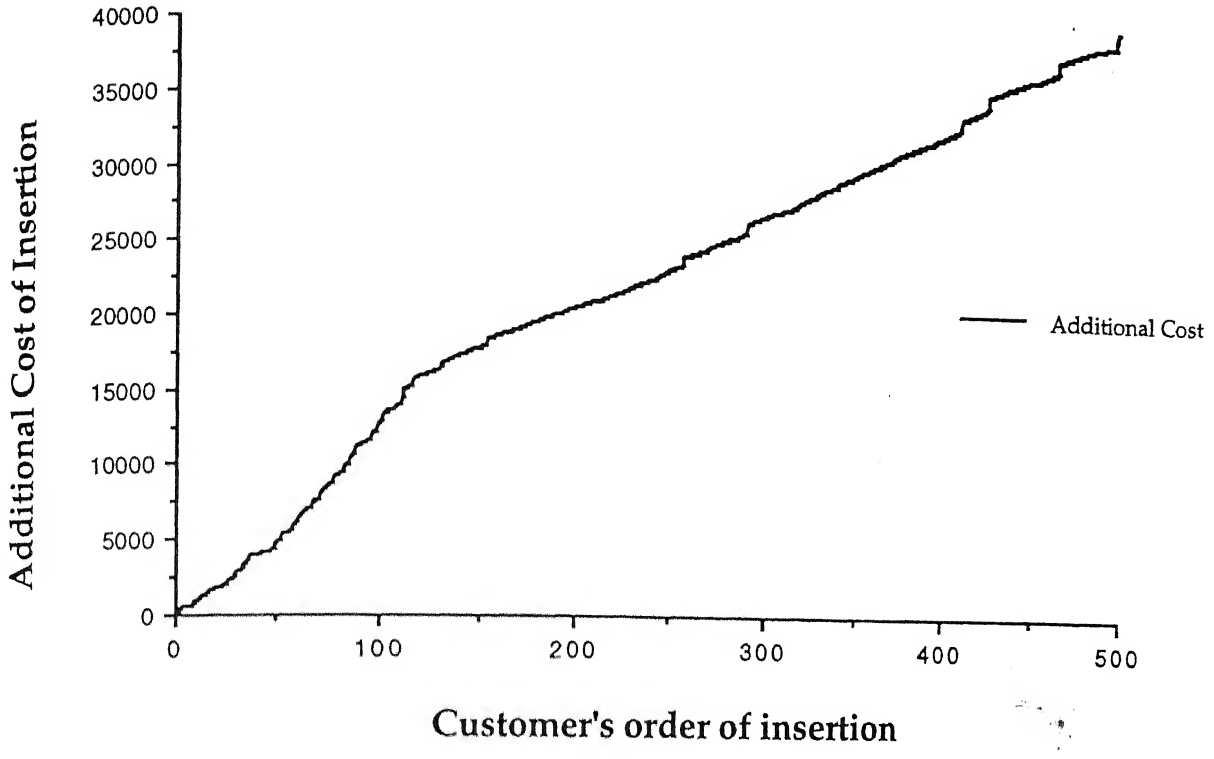
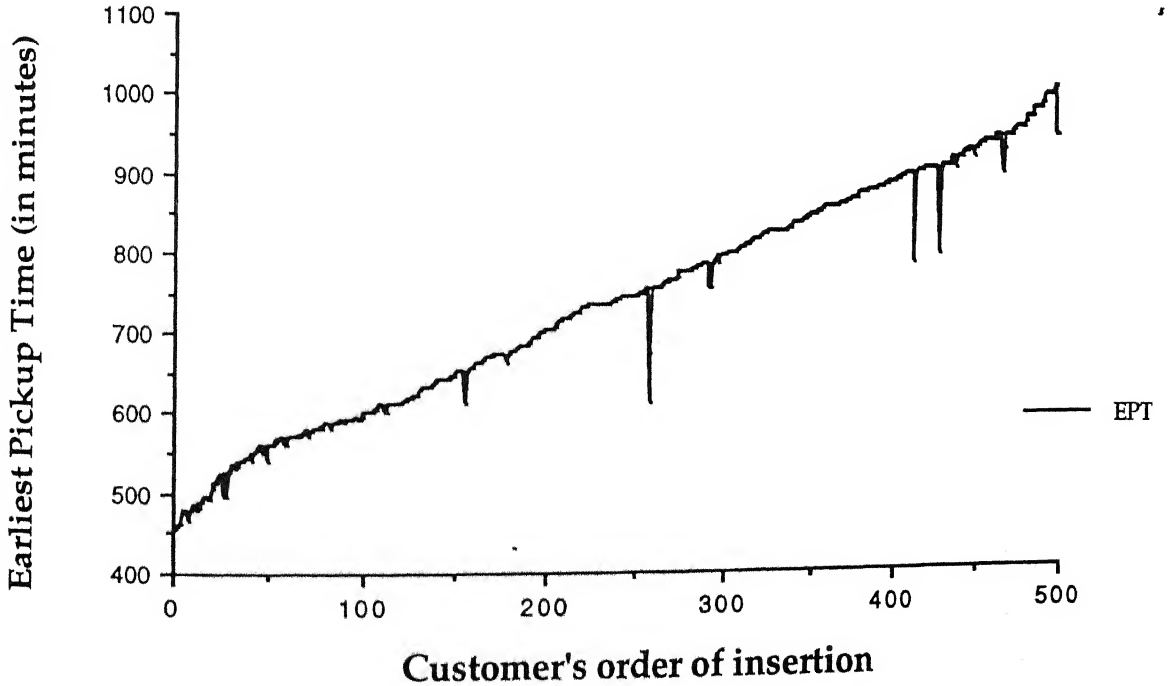


Figure 3.55

Earliest pickup time of customers as the order of their insertion for immediate refilling of pool



customer's insertion for periodic refilling of pool

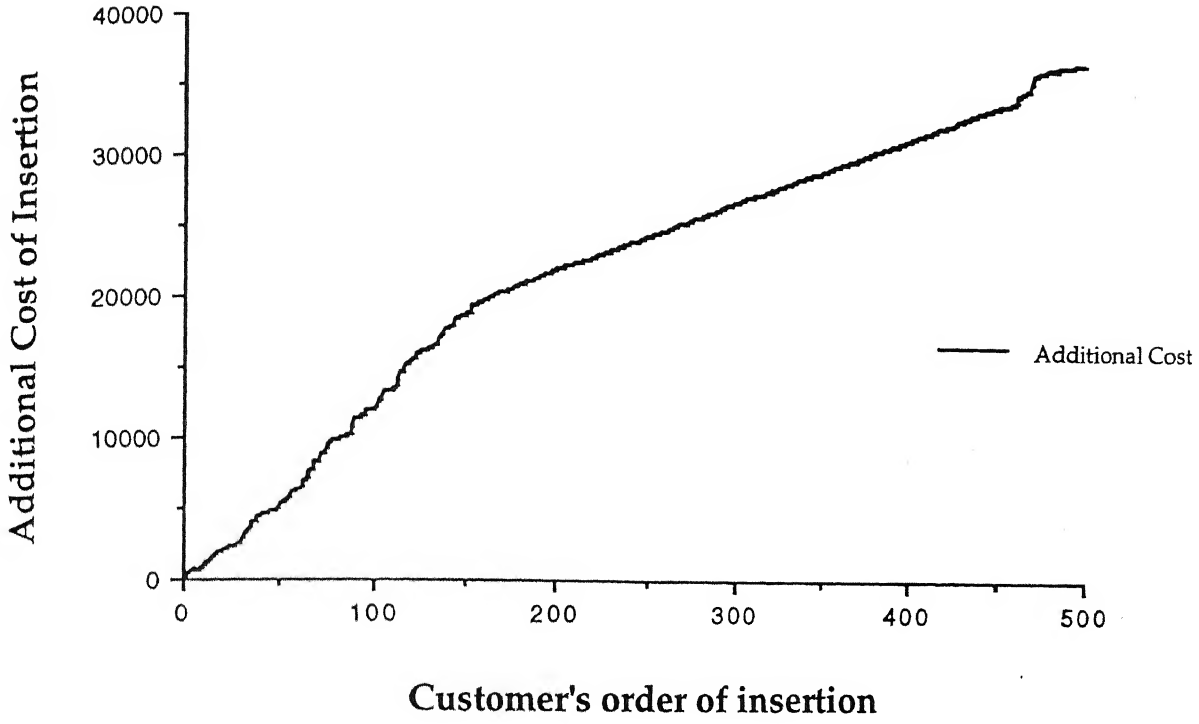


Figure 3.57

Earliest pickup time of customers as the order of their insertion for periodic refilling of pool

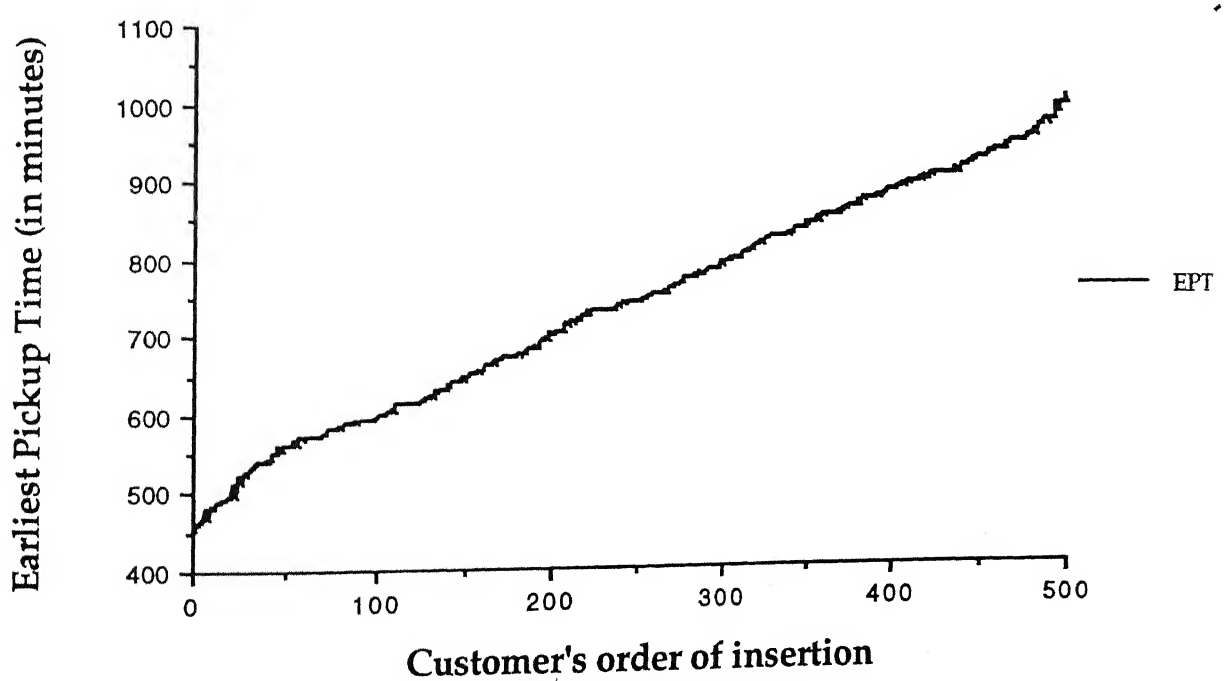


Figure 3.58

Graph dipicting effect on execution time by changing number of requests, pool size and pool refilling strategies

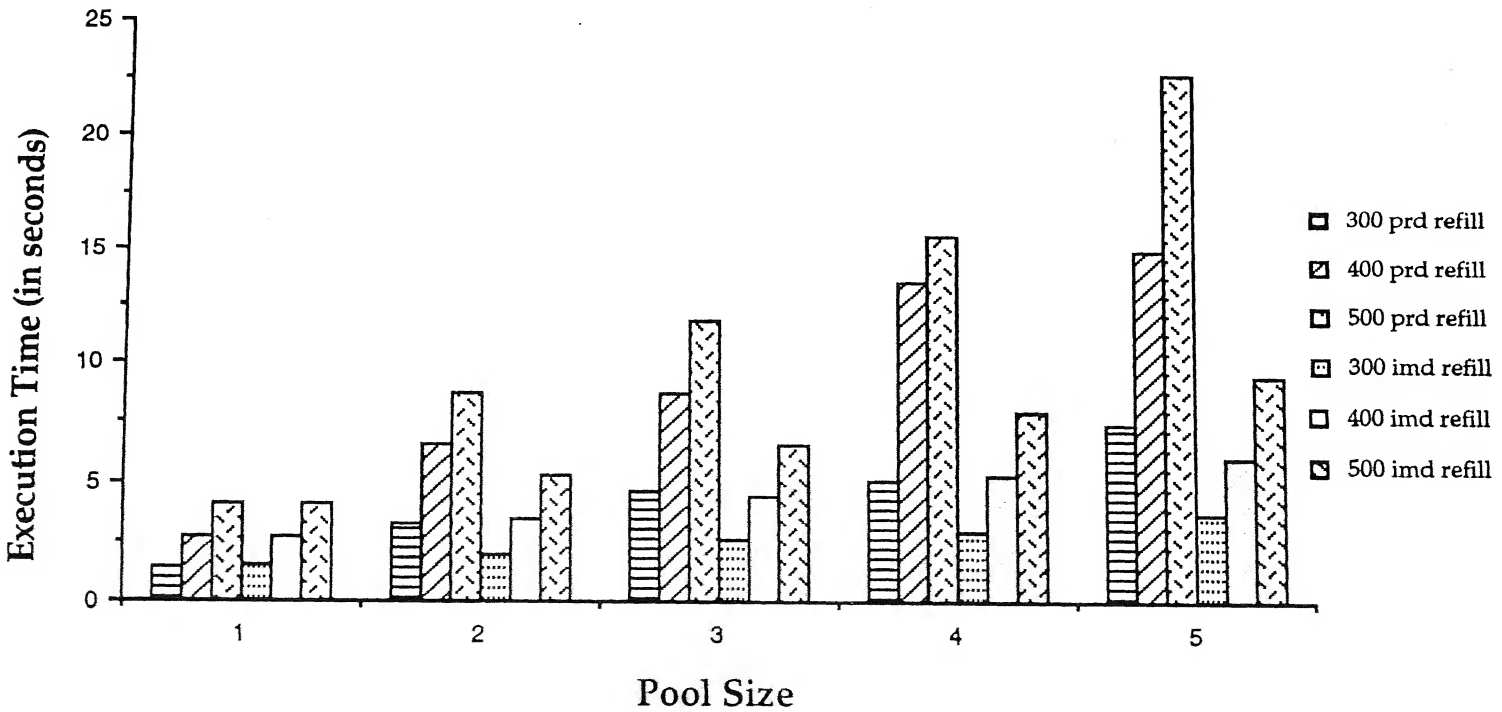


Figure 3.59

can be computed to be (d)(d-1)/2. Since there are N/d batches of customers to be processed, the total number of updates required for the periodic-refill strategy is (d-1)N/2 which is only half of what s required for the immediate-refill strategy.

Another somewhat minor contributing factor to the significant increase of computation time under the immediate-refill strategy is the effort involved in the selection of the best customer from a pool of candidates. Since the pool is always full when a selection is made under the continuous-refill strategy. ADARTW has to perform d comparisons in order to choose the minimal cost insertion. Whereas under the periodic refill-strategy, an average of d/2 comparisons is required each time. The total difference between the two strategies is again proportional to (d/2).N.

So far we have examined empirically the computational efficiency of ADARTW. It is also important that we study the computation complexity of ADARTW from a theoretical point of view. Normally, the computation complexity of an algorithm can be determined by the number of elementary operations required in the process. For ADARTW, the number of possible insertions for a customer can go as high a N^2 (worst case scenario when all N customers are assigned to one vehicle). For each insertion a feasibility check is required. Despite the fact screening test we have designed to cut down the computation time, a linear search has to be performed through the list of vehicle worst-schedules which can take as many as cN comparisons (c is some constant). So far each customer, $O(N^3)$ elementary operations are necessary for the worst case scenario. Since there are N customers, the computational complexity of ADARTW is $O(N^4)$. When considering multiple candidates in ADARTW, the extra effort involved is also of order N^3 (if $d << N$) for both pool-refilling strategies. So the computational effort

the same time.

Empirically, from Figure the computation time of ADARTW does not seem to grow as fast as $O(N^4)$ as is suggested theoretically. This is due to the fact that $O(N^4)$ is just the worst case and such a worst case rarely happens in practice. For example, in practice, the N customers are more likely to be evenly distributed among m vehicles, Thus, the computation effort of processing one customer reduces to the order of $m.(N/m)^3$. The efficient screening tests introduced in section 3.5 will also normally reduce the computation time by detecting infeasibilities early in the process.

CHAPTER 4 RANDOMIZATION

4.1 INTRODUCTION

After implementing advance request the next question which came to our mind was how good is it to always select the customer with the least cost of insertion? Can we select the other customers with higher insertion costs? If we do so then how is it going to affect the overall quality of constructed schedules?

The intuition behind to insert the customer with least insertion cost was to maintain some greediness while construction of feasible schedules. In this method while making decision regarding insertion we consider the insertion possibilities of only those customers which we have selected in customer pool. Hence while performing insertions we are not concerned about the cost of future insertions. Trying out few insertions, which are not the best in terms of objective function at a particular instant, may generate routes such that the incremental cost of future insertions may be less.

To explore the above possibilities we have tried some randomization approaches in the hope that we may find some of schedules better than the original one.

In Section 4.2, we will describe two different strategies to generate the feasible schedules which involves certain degree of randomization but with bias towards better choice in the selection of customer to be inserted. In Section 4.3 we will discuss unbiased randomization approach. In Section 4.4 we will discuss computational results of the various randomization approaches. Section 4.5 discusses a model which will simulate that how exactly normal

taxi will serve this geographic area. Finally, in Section 4.6 we have discussed the savings achieved by schedules generated by the ADARTW algorithms with the normal taxi service.

4.2 BIASED RANDOMIZATION

In this approach, while constructing routes (i.e., allocating the advance request among vehicles) we maintain certain level of greediness in picking up the possibility of insertion among p best possibilities that exists for customers in customer pool among fleet of vehicles. This is explained in the following example.

Consider for example, there are 10 vehicles and 100 requests are to be assigned among the 10 vehicles. Firstly, all advance requests are sorted in their EPT order. Now assuming that customer pool size is 1, which means that we will explore the possibility of insertion for only one customer among the available vehicle fleet at a time. We now select a customer having least EPT among the remaining unconsidered customer. Now say there exists 30 possibilities of inserting this customer in all 10 vehicles.

Out of these 30 Possibilities we keep only p best possibilities of insertion. Here p is a parameter which can be varied to generate different solutions. Now we generate a random number between 1 and p. For randomly generated number x, we will perform the x^{th} best insertion for selected customer unlike the best one as in the case of advance request algorithm. Thus each value of p will generate different solutions.

The random number generated between 1 and p can have equal probability or they may generated with different probability.

For biased randomization random number are generated in such a way that the lower the number is, the higher the probability of its generation. In this way the greediness is maintained in the selection of an insertion possibility. The more the biasing towards generation of smaller number the higher will be the greediness.

Computational results will be presented with keeping customer pool size equals to one and varying p, in Section 4.4. We will call this strategy as *Fixed Pool Biased Randomization*.

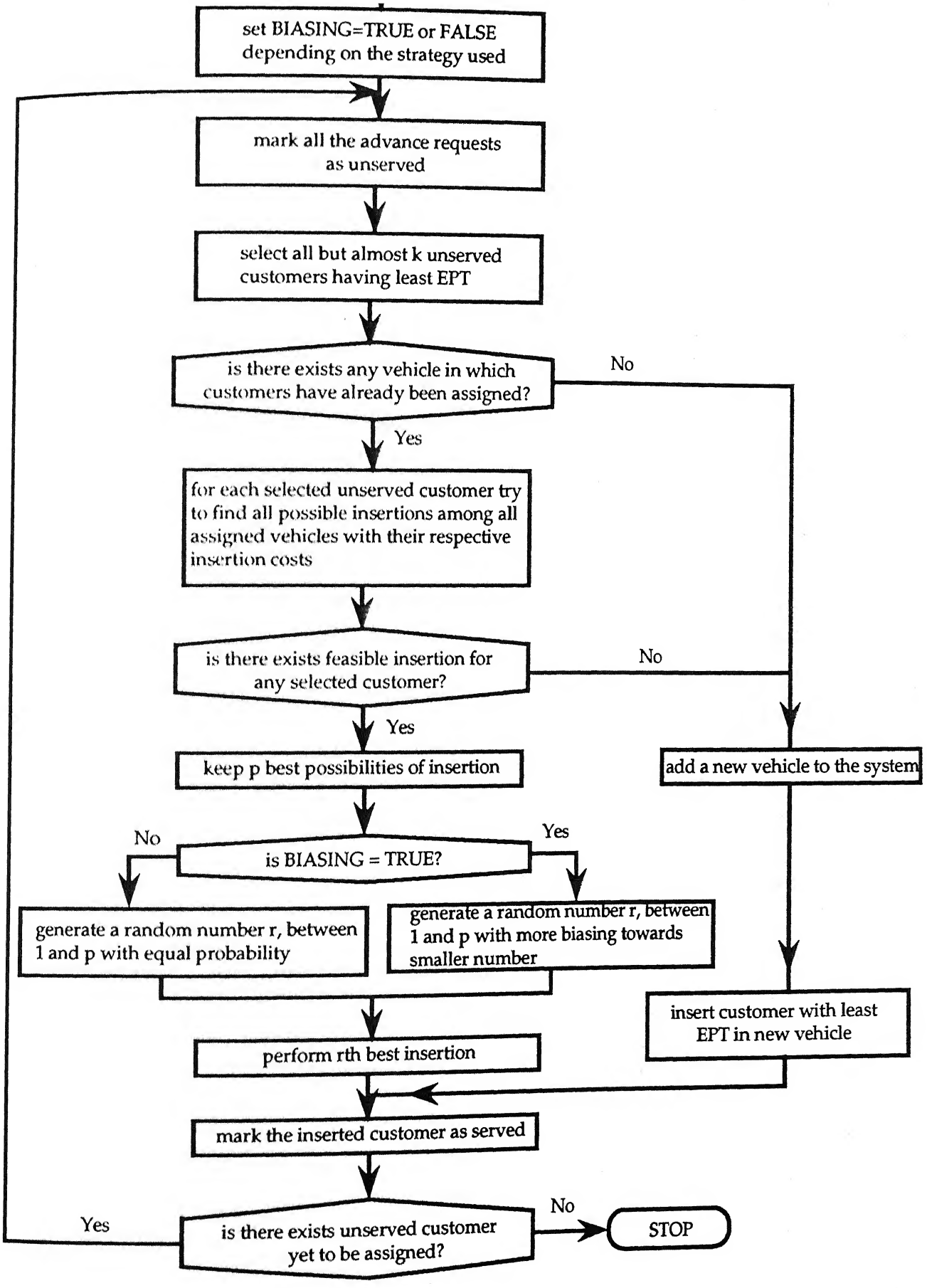
Keeping customer pool size equal to one means that the order of insertion of customers are fixed. Thus to have some degree of randomization in the order of insertion we can have a varying size of customer pool as p is varied. We will call this strategy as *Varying Pool Biased Randomization*. Computational results of this strategy is presented in Section 4.4.

4.3 UNBIASED RANDOMIZATION

In this approach while we are allocating advance requests among vehicles we give equal chance to p best insertion possibilities, among the vehicles, that exists for the customers in customer pool.

It is almost same as Biased Randomization but with the only difference in generating the random number. Here random number x, between 1 to p, is generated with equal probability. Thus there is no bias towards better solution. The customer is inserted according to x^{th} best way.

In unbiased randomization we again tried two approaches similar on the lines as explained in previous section. We will call the first approach as *Fixed Pool Unbiased Randomization* . In this customer pool is fixed in size and equal to one. For different values of p results have been taken.


set BIASING=TRUE or FALSE depending on the strategy used
mark all the advance requests as unserved
select all but almost k unserved customers having least EPT
is there exists any vehicle in which customers have already been assigned?
No
Yes
for each selected unserved customer try to find all possible insertions among all assigned vehicles with their respective insertion costs
is there exists feasible insertion for any selected customer?
No
Yes
keep p best possibilities of insertion
add a new vehicle to the system
No
is BIASING = TRUE?
Yes
generate a random number r, between 1 and p with equal probability
generate a random number r, between 1 and p with more biasing towards smaller number
insert customer with least EPT in new vehicle
perform rth best insertion
mark the inserted customer as served
No
Yes
is there exists unserved customer yet to be assigned?
STOP

Another approach is call as *Varying Pool Unbiased Randomization*. In this approach size of customer pool as well as p is varied.

Computational results will be presented in Section 4.4 for above approaches. Flow chart for all the randomization strategies described above is given in figure 4.1.

4.4 COMPUTATION RESULTS FOR RANDOMIZATION

We have divided the discussion of computational results into two parts. Part one discusses the various randomization approaches described above. In part two, we have compared the performance of best of the randomization strategy with the taxi simulation. For this purpose we have made a taxi simulation model, which will also be described in the part two.

4.4.1 Fixed pool unbiased randomization

We have investigated the effects of increasing parameter p on the objective function for five different instances of 500 customers. Each instance for which $p \geq 2$, we have plotted best objective function obtained from ten runs with different values of seed to random number generator. The seed values used are as follows :

Value of p	Seed Range
2	11 to 20
3	21 to 30
4	31 to 40
5	41 to 50

It can be easily be noticed that p = 1 and customer pool size = 1 is same as immediate pool refilling with customer pool size equals one. As discussed in section 3.7.5, we know that for immediate refill pool size = 1 provides the best schedule. Thus in figure 4.2 p = 1 corresponds to the best solution obtained from ADARTW with immediate pool refill.

As we increase p from 2 to 5, the best objective function obtained deteriorates continuously (figure 4.2). Here, since pool size is one, this means that selection of inferior possible way of inserting a customer will adversely affect the quality of the generated schedules. This suggests that biasing is indeed important for generation of better schedules.

4.4.2 Variable pool unbiased randomization

In this experiment, we have tried to eliminate the myopic nature of previous strategy by considering more than one customer at a time. Also, we have kept the value of p equal to the customer pool size so as to give chance to each customer in customer pool.

Our observation in figure 4.3 suggests that increasing p and pool size also will affect objective function adversely. This may be due to two reasons. Firstly, due to lack of biasing towards better insertion possibility. Secondly, as discussed in the Section 3.7.5, that for immediate refill strategy higher value of pool size will result in out of EPT order insertions, and which in turn will affect the schedules negatively.

4.4.3 Fixed pool biased randomization

As we have observed in the case of unbiased randomization the schedules generated were not good. We will now present the results of fixed pool biased randomization for the following values of biasing :

seed by fixed pool unbiased randomization strategy

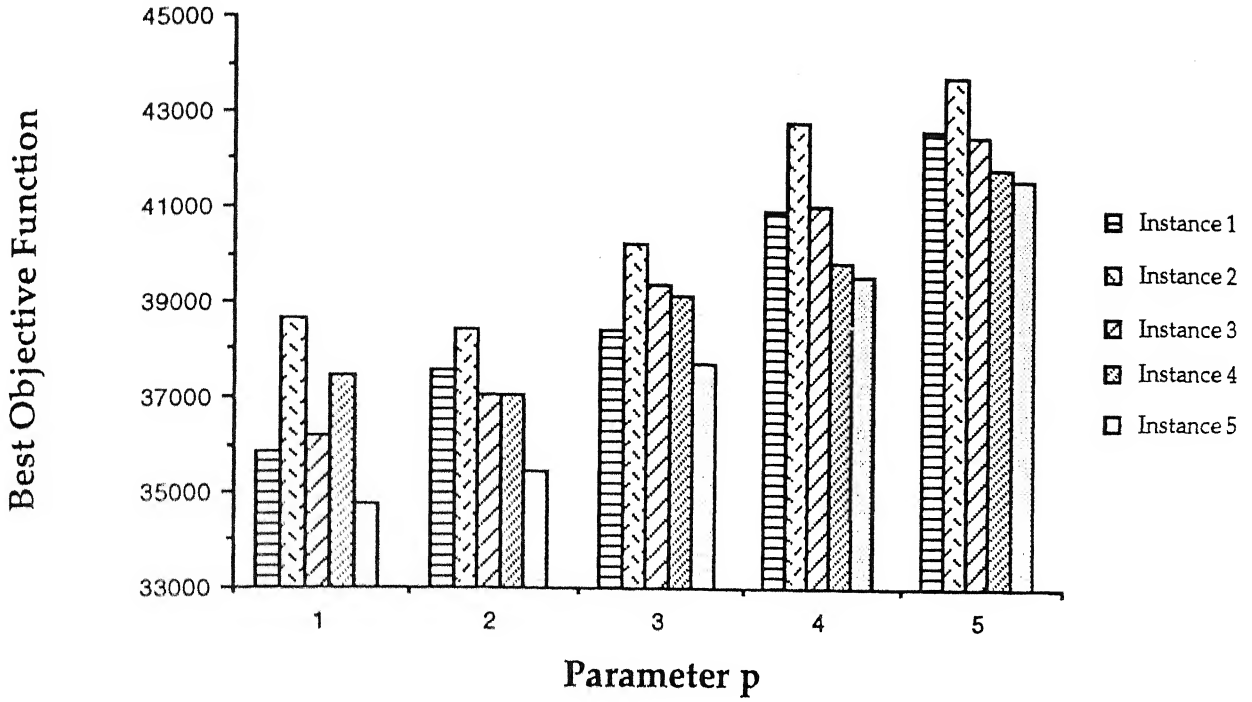


Figure 4.2

Best objective function value found in 10 runs with different seed by varying pool unbiased randomization strategy

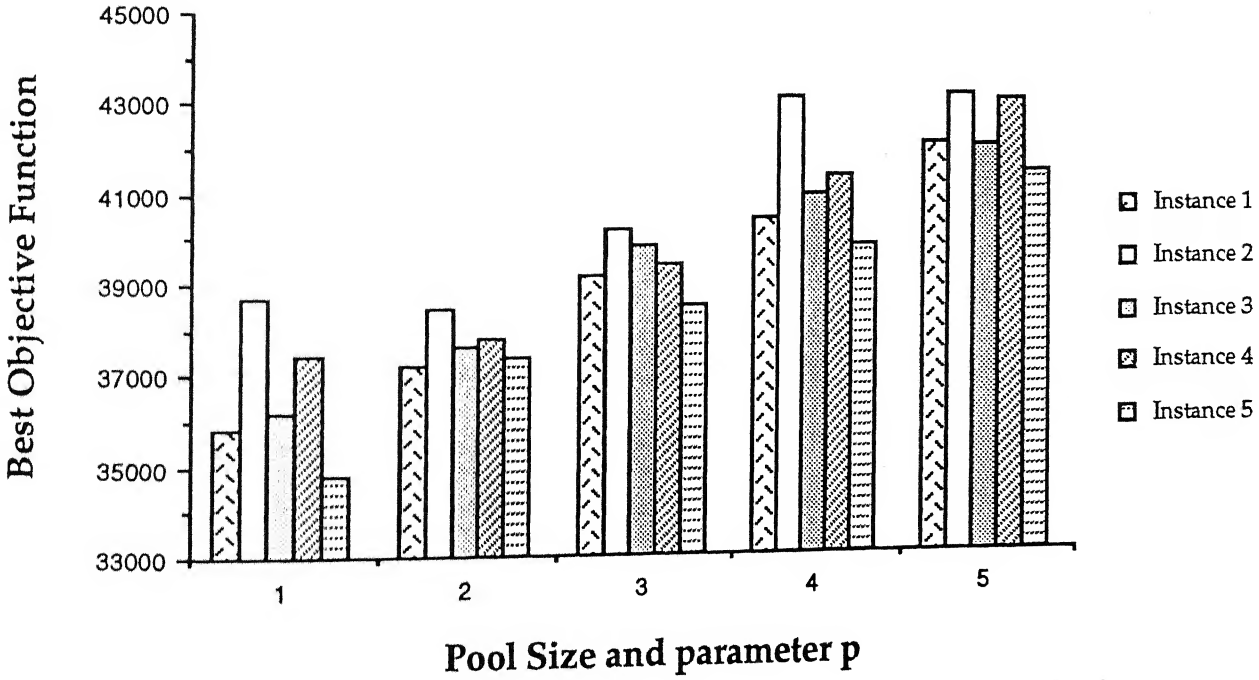


Figure 4.3

Value of p	Bias % for best possibility of insertion	Bias % for 2nd best possibility of insertion	Bias % for 3rd best possibility of insertion	Bias % for 4th best possibility of insertion	Bias % for 5th best possibility of insertion
2	75%	25%	-	-	-
3	70%	20%	10%	-	-
4	65%	15%	10%	10%	-
5	65%	15%	10%	10%	5%

We, in the figure 4.4, have plotted the best objective function value obtained from 10 runs for the seed value given in section 4.4.1. It can be easily observed from the figure 4.4 that for values of $p > 3$ the quality of the schedules generated deteriorates. Which implies that to generate good schedules we should keep only 2 or 3 best insertion possibilities of a customer and the occasionally introduce the worse possibilities. Improvement obtained for some instances are in the range of 8% to 10%.

4.4.4 Variable pool biased randomization

In this strategy we have used the seeds given in section 4.4.1 for performing 10 runs. The value of biasing used is also the same as what we have used for fixed pool biased randomization.

As the value of p and pool size is increased from 2, objective function is affected adversely (figure 4.5).

Thus it seems that value of $p = 2$ for both the biased strategies are the only promising ones. Thus we have chosen $p = 2$ and pool size = 1, and $p = 2$ and pool size = 2 as the most promising values of the parameter to be used in the biased randomization strategies. In the following section we will describe the effect of changing of bias on above two sets of parameter values.

Best objective function value found in 10 runs with different seed by fixed pool biased randomization strategy

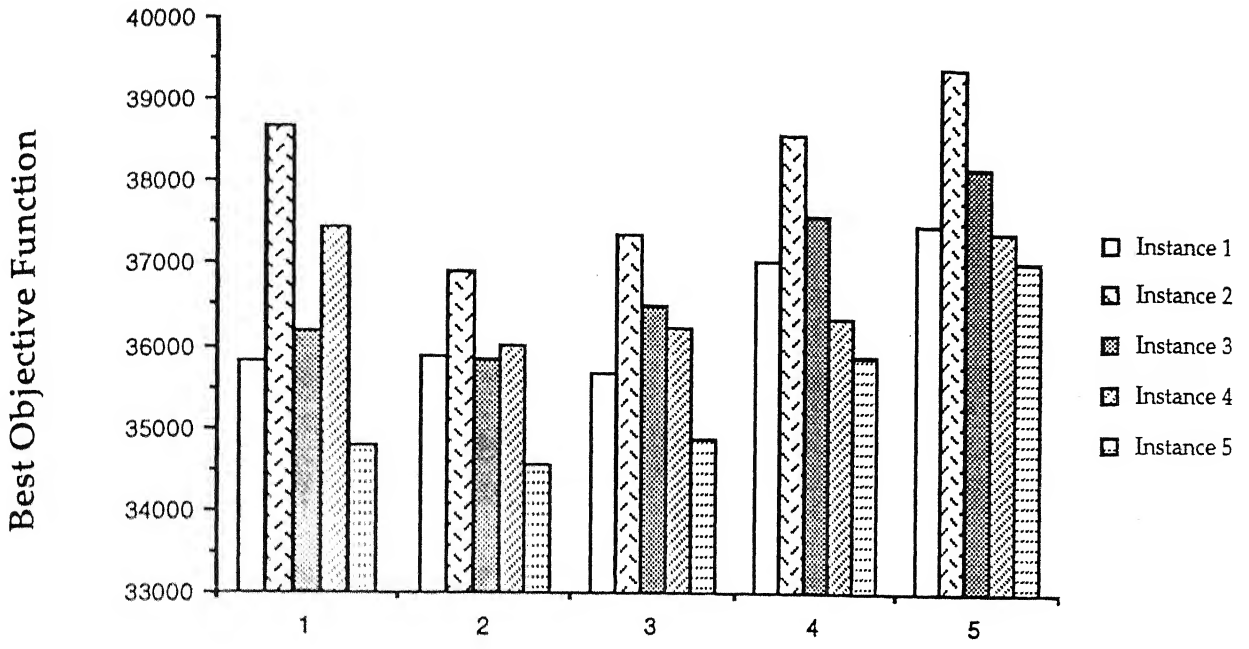


Figure 4.4

Best objective function value found in 10 runs with different seed by varying pool biased randomization strategy

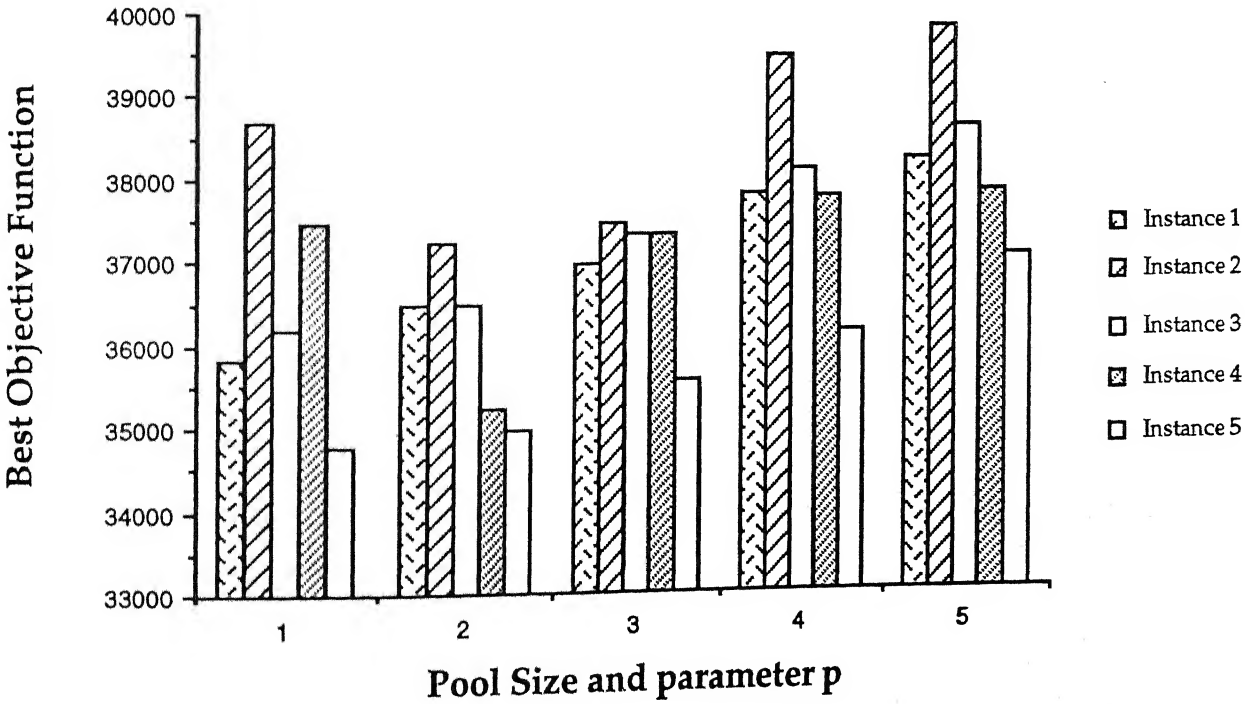


Figure 4.5

4.4.5 Effect of changing bias on the objective function

We have chosen two different instances to study the effect of changing the value of bias on the objective function. Bars corresponding to 0% in all the figures from figure 4.6 to figure 4.9 represents immediate pool refill with pool size = 1 to compare the performance with the biased randomization approaches. We have experimented with four different values of seed.

As evident from these figures that for low bias the schedules are not good, but once bias value is in the range of 75% and higher we found significant improvements in objective function values (approximately 8% to 10%). Thus it seems that biasing is very important. Both strategies give the improvements, but since pool size = 1 will take lesser amount of CPU time, it will be more preferable.

4.5 TAXI SIMULATION

After making all algorithms for ADARTW now the most important issue is to determine it's benefits over the ordinary taxi service? This benefit is very important to support the idea of ADART. So to compare the performance of the ADARTW schedules and ordinary taxi schedules, we made a simulation model for ordinary taxi. For ADART service there exists both types of requests i.e., advance requests and immediate requests. But for the taxi simulation, all the requests are immediate requests. The flow chart for taxi simulation is given in figure 4.10.

In this we INTERVAL is taken to be 15 minutes. This means that after every 15 minutes allocation is made at the end of the schedule block among the vehicles. Customers are assigned among the vehicles according to the time at which request is made. We have compared taxi service with the advance

biasing for fixed pool biased randomization strategy in which pool size is 1 and parameter p is 2

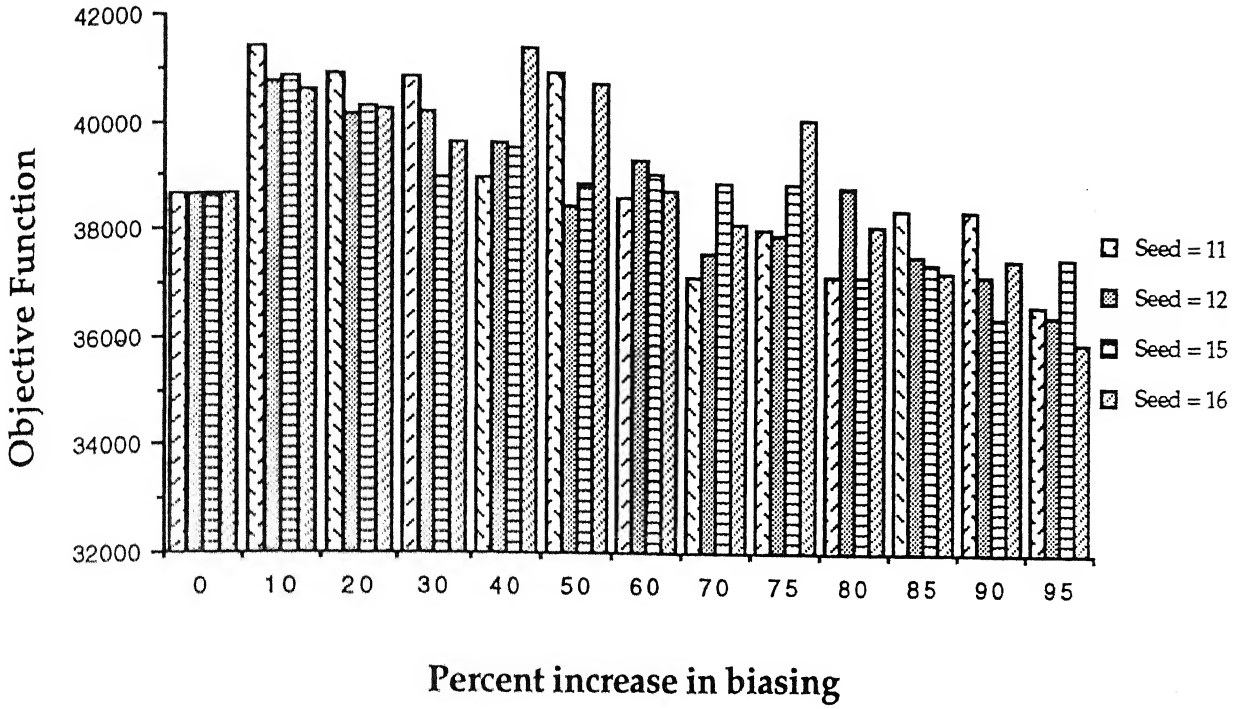
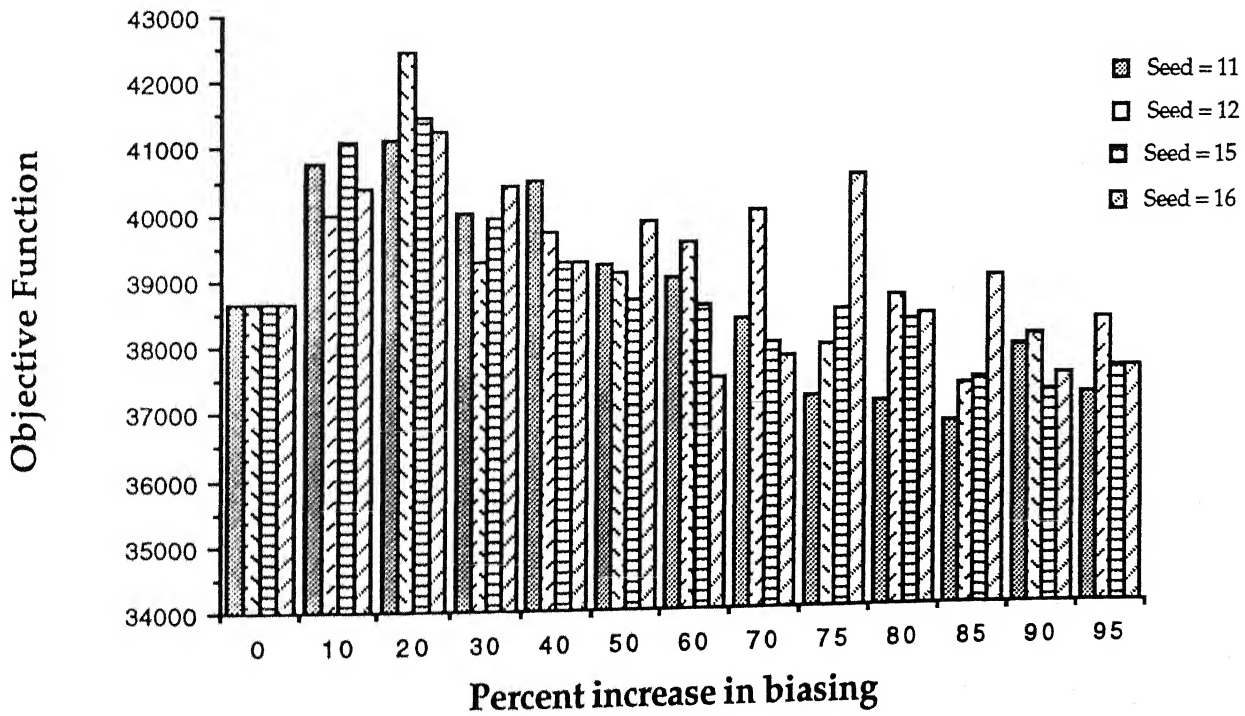


Figure 4.6

Change in objective funtion with increase in degree of biasing for variable pool biased randomization strategy in which pool size and parameter p both are 2



biasing for fixed pool biased randomization strategy in which pool size is 1 and parameter p is 2

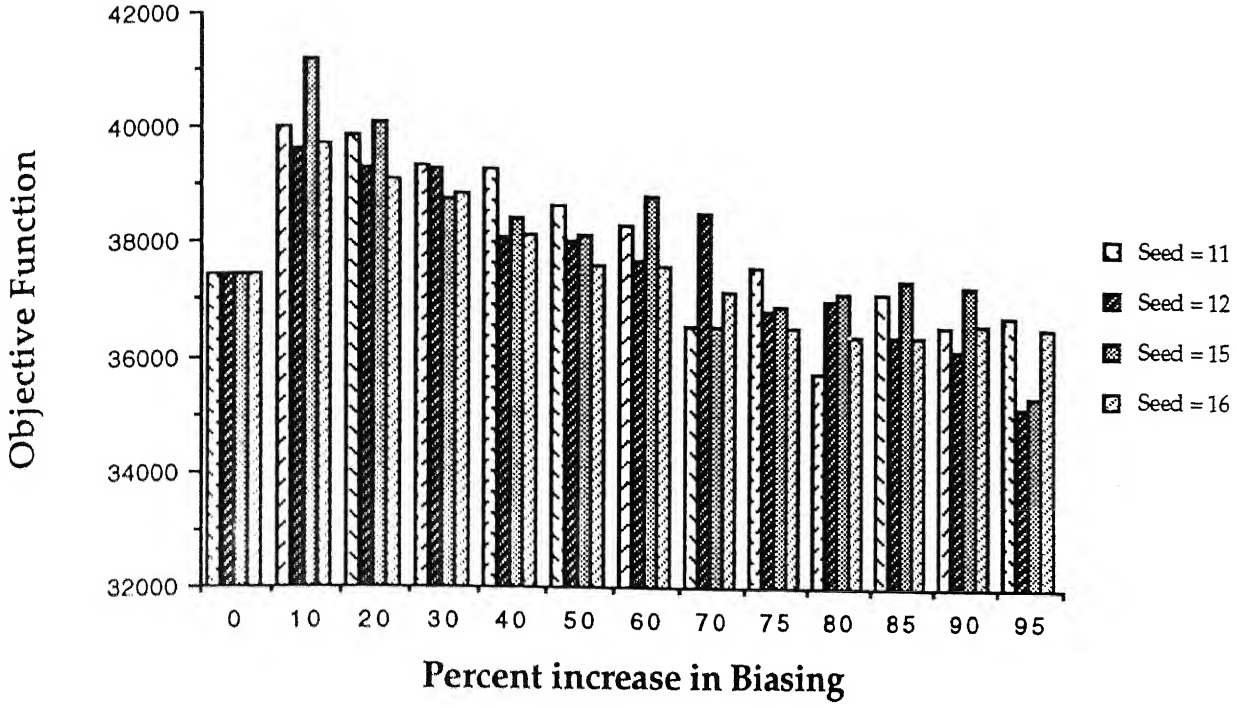


Figure 4.8

Change in objective funtion with increase in degree of biasing for variable pool biased randomization strategy in which pool size and parameter p both are 2

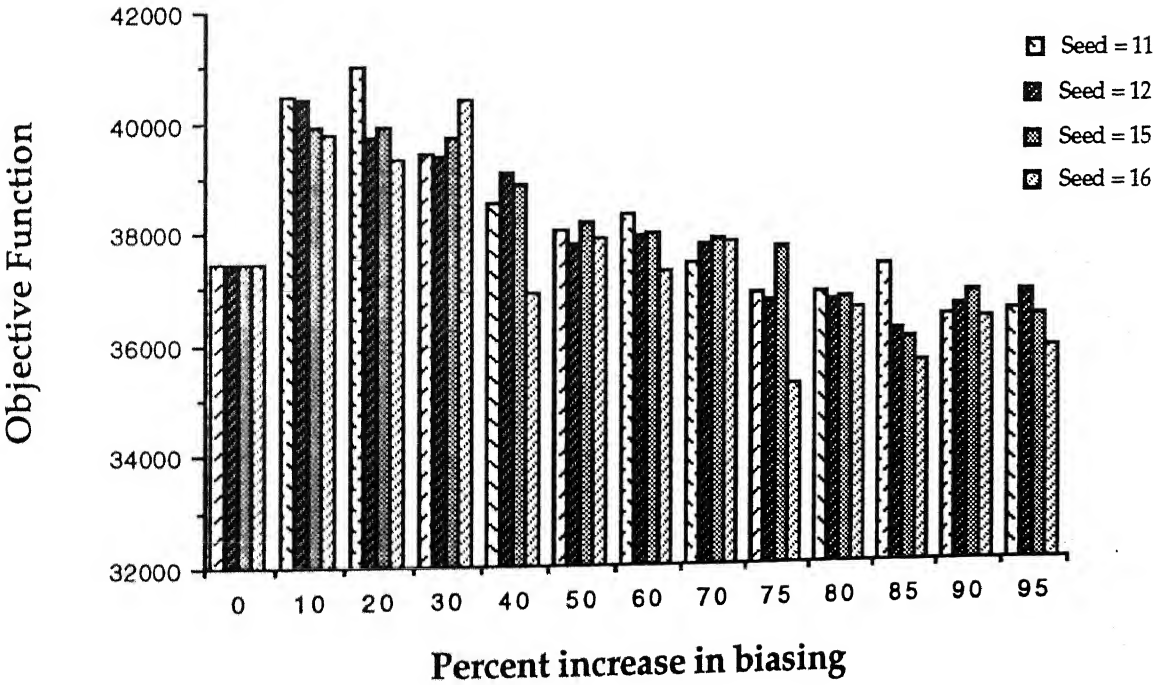


Figure 4.9

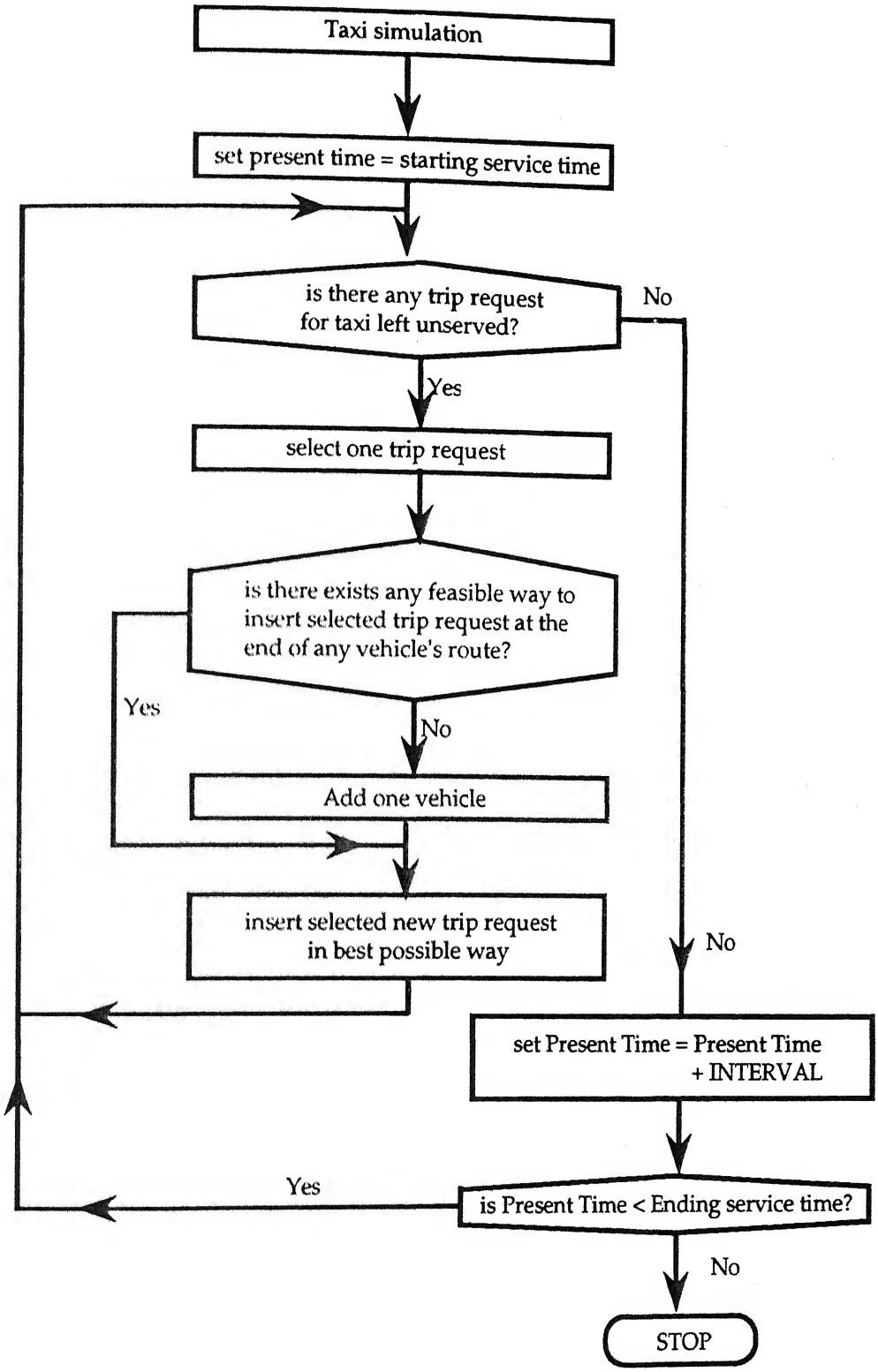

Taxi simulation
set present time = starting service time
is there any trip request for taxi left unserved?
No
Yes
select one trip request
is there exists any feasible way to insert selected trip request at the end of any vehicle's route?
Yes
No
Add one vehicle
insert selected new trip request in best possible way
No
set Present Time = Present Time + INTERVAL
Yes
is Present Time < Ending service time?
No
STOP

Figure 4.10

request version of ADART i.e. ADARTW schedules. As mentioned in the section 3.2.1 that ADART system should perform better for many-to-few type of requests as compared to normal taxi service. The computational results for this comparison is also presented in the following section.

In private taxi, the customer do not share taxi with others. So the customer can be added only at the end of the schedule but in ADART system customers can share taxi with others so it promises better productivity of vehicle resources.

4.6 Computational Results For Taxi Simulation

In section 3.2.1 we have described about various types of DART service. Many-to-few type of service seems to be more attractive form of option as this would result into reasonably higher utilization of the vehicle resources. We have generated customer subscription list such that it has varying amount of mix of MTF customers as 0%, 25%, 50%, 75% and 100%.

We have obtained the best solution out of 5, for different runs for biased pool randomization with pool size and p equal to 2. Results for different statistics are analyzed in the following paragraphs.

As shown in the figure 4.11 that the deviation decreases continuously with the increase in the percent of MTF customer for randomization approach. Consider for example, say, location x is a theater in a city. All of MTF type customers will be delivered at the same time to location x. These customers will be of desired delivery type as they have to reach this location at the time specified by them. Also, at location x there will be customers (after end of the show), which are pickup specified, wanting to leave x at time specified by them. Since here desired pickup time for customer wanting to leave the place

will be the same as the desired delivery time of the customers coming to this place. If vehicle reaches in time there will be zero deviation for all the customers coming and leaving the place x. Thus as the percent of MTF type of customer increases, the deviation reduces.

From figure 4.12 it can be seen that the excess ride time in case of taxi service is zero, but for randomization approach it is 57% higher than DRT.

Figure 4.13 shows that total vehicle time decreases for randomization approach as the percent of MTF customers increase. This is due to the higher sharing of the vehicles in the case of higher MTF customers. Also, number of vehicles required reduces to almost half (from 60 to 30) as seen in the figure 4.14 for randomization approach. For 100% MTF customers number of taxi required is almost four times higher than required for randomization approach. As shown in the figure 4.15, vehicle utilization for randomization approach remains constant to 96%, but for taxi simulation it reduces to almost 65%. This is due to lot of unproductive slack of vehicles as there exists lot more vehicle in the case of taxi simulation. Vehicle productivity rises to 6 for randomization approach with the increase in the MTF customer percent (figure 4.16).

Comparison of deviation from desired pickup or delivery time with the change in percent mix of many-to-few type of customers for randomization strategy

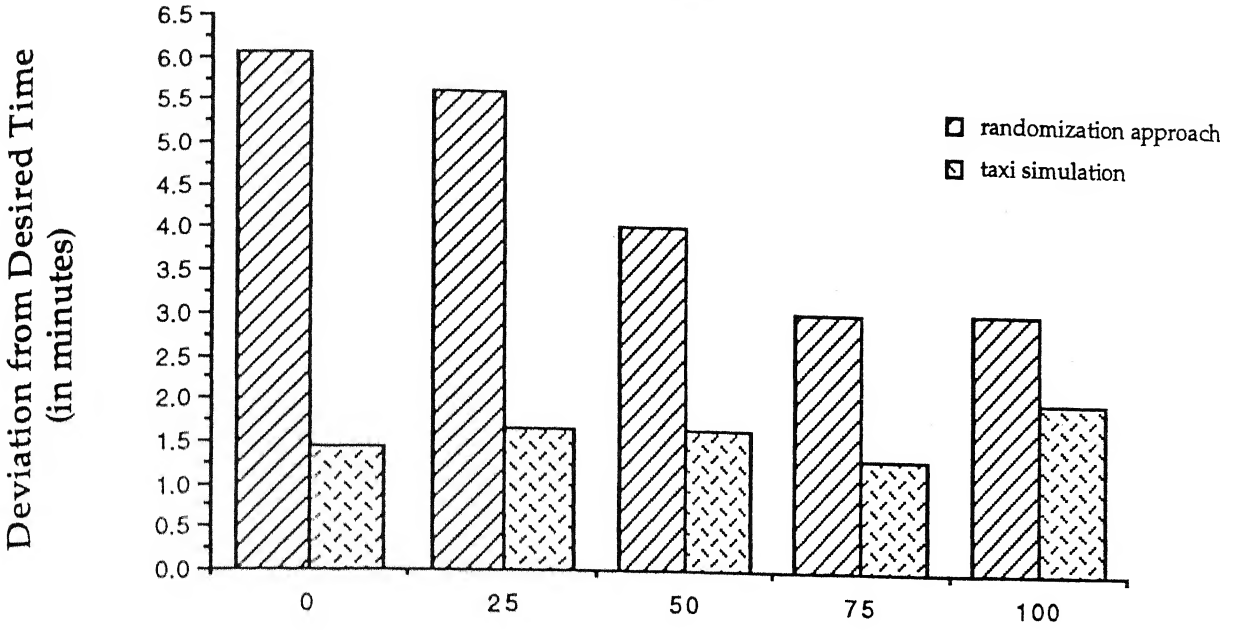
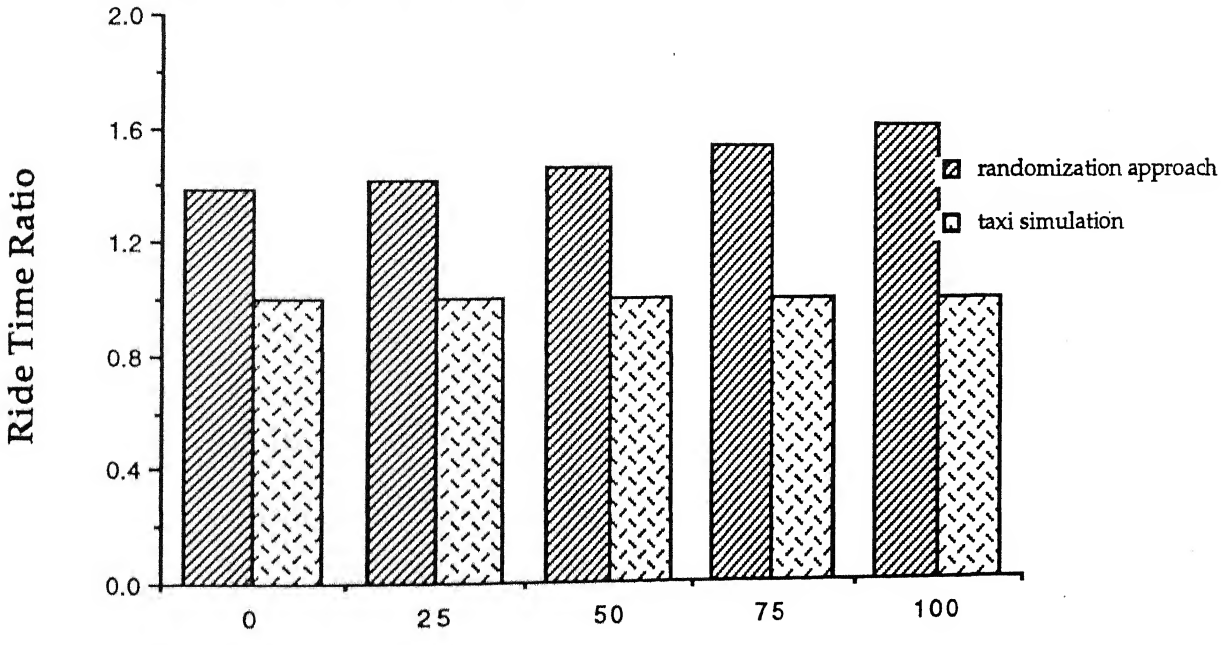


Figure 4.11

Comparison of ride time ratio for taxi simulation with the randomization approach for serving 1000 requests with change in the percent mix of many-to-few type of customer



Comparison of vehicle time for taxi simulation with the randomization approach for serving 1000 requests with change in the percent mix of many-to-few type of customer

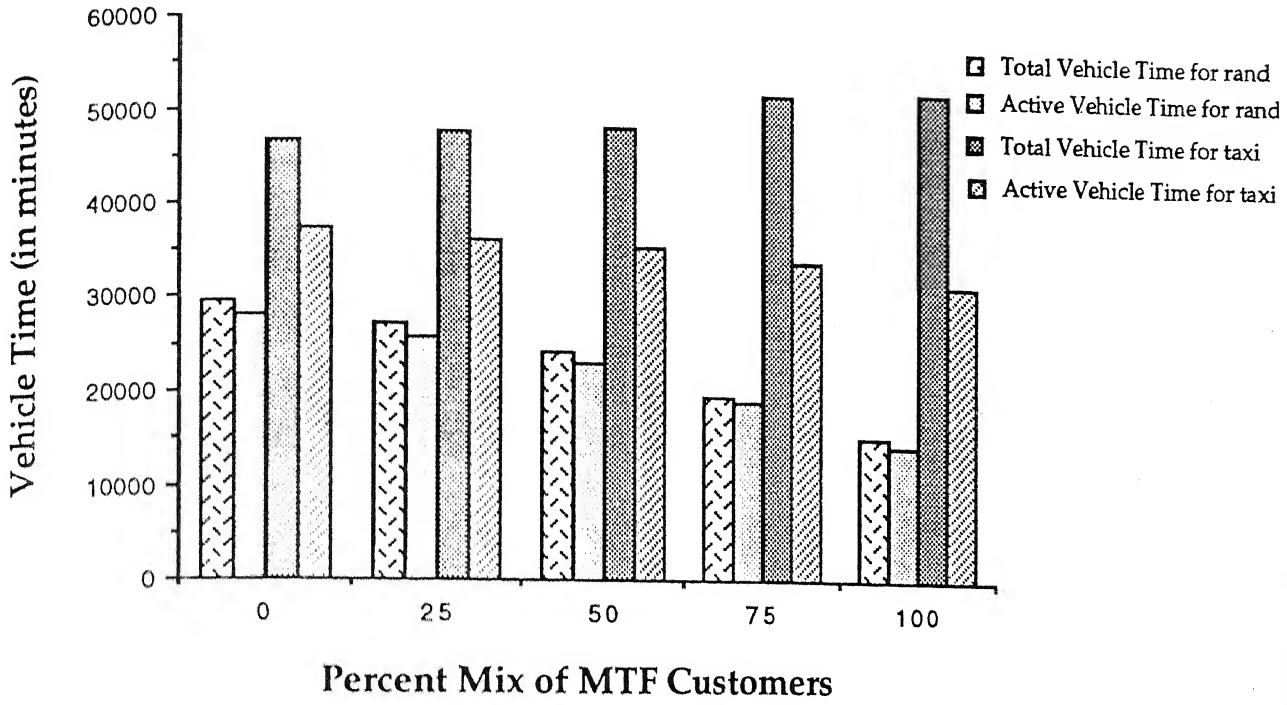
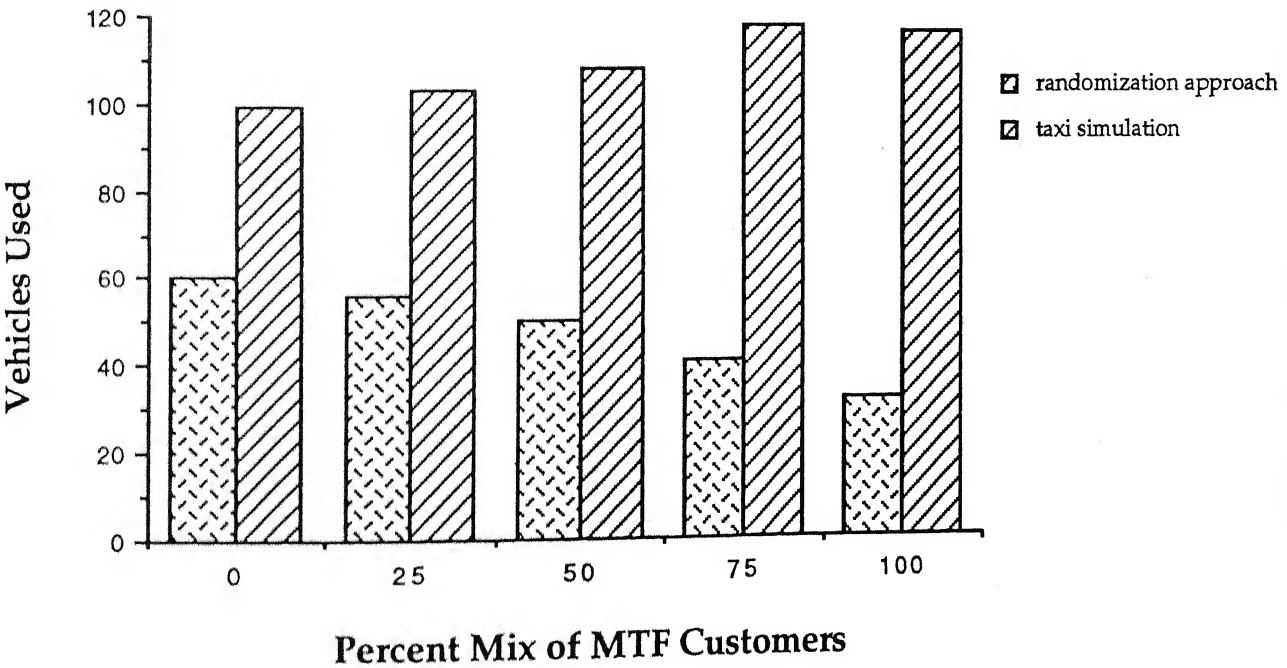


Figure 4.13

Comparison of vehicles used for taxi simulation with the randomization approach for serving 1000 requests with change in the percent mix of many-to-few type of customer



Comparison of vehicle utility for taxi simulation with the randomization approach for serving 1000 requests with change in the percent mix of many-to-few type of customer

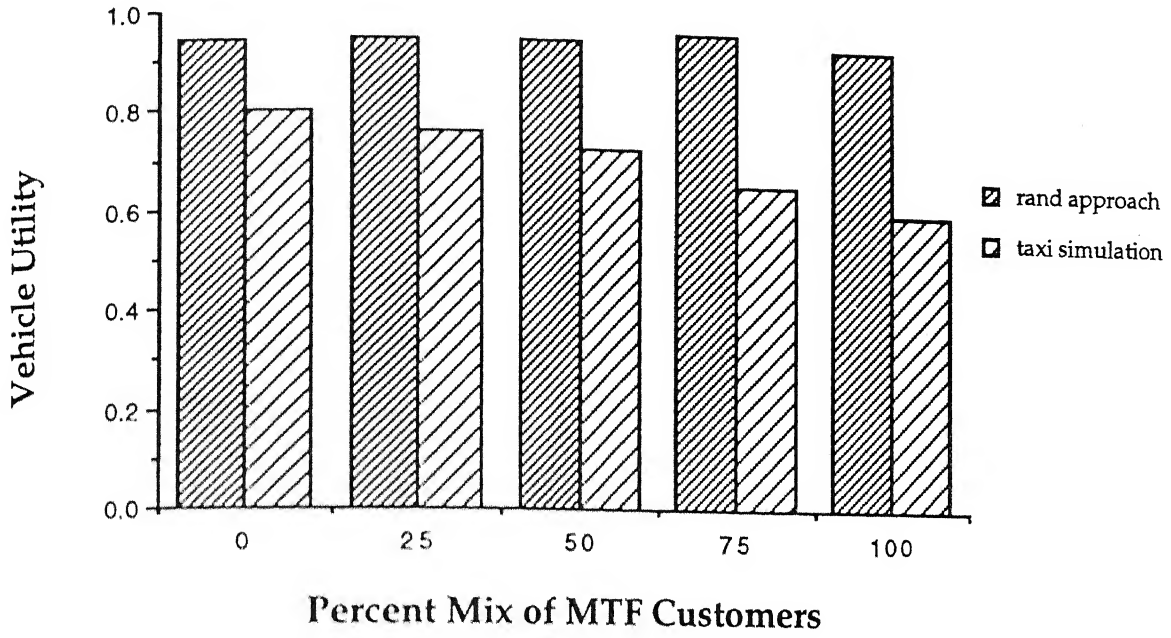
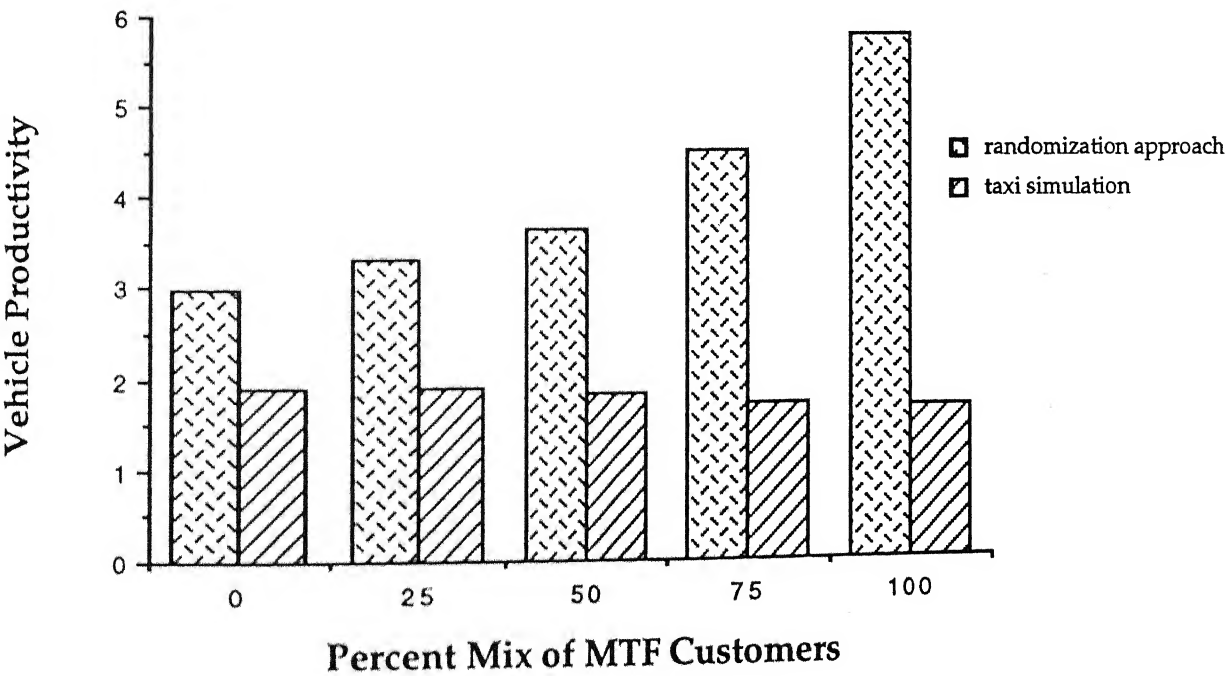


Figure 4.15

Comparison of vehicle productivity for taxi simulation with the randomization approach for serving 1000 requests with change in the percent mix of many-to-few type of customer



4.7 CONCLUSIONS

From the above results we can conclude that this system saves vehicle resources to a phenomenal extent as compared to normal taxi service to serve same set of customers. Due to these savings it provides a better, cheaper and reliable alternative to a taxi system, particularly when this system targets MTF type of service.

MTF type of sevice has a tremendous potential in any big city. Thus we think that approaches developed in this thesis can be implemented commercially.

I have limited the scope of my work to the generation of schedules for advance trip requests. Deepak [9] has further extended this work by developing several improvement schemes for schedules generated by the approaches developed in this thesis. He has also implemented the algorithms to modify schedules based on new trip requests and cancellations.

REFERENCES

[1] Wilson, Nigel H.M., Weisseberg, Hauser, "Advanced Dial-A-Ride algorithms research project: final report", *Department of Civil Engineering , M.I.T. Report* R76-20, (1976).

[2] Wilson, Nigel H.M., Joseph M. Sussman, Ho-Kwan Wong, "Scheduling algorithms for Dial-A-Ride system", *Urban Systems Laboratory*, USL TR-70-13, M.I.T. (1971).

[3] Psarafits, H.N., "A dynamic programming solution to the single vehicle many-to-many immediate request Dial-A-Ride problem", *Transportation Science*, 14, 130-154 (1980).

[4] Psarafits, H.N., Tharakan, "A dynamic programming approach to Dial-A-Ride problem : An extension to the multi-vehicle case", *Department of Civil Engineering , M.I.T. Report* R79-39, (1979).

[5] Psarafits, H.N., " An exact algorithm for the Single vehicle many-to-many Dial-A-Ride problem with time windows", *Transportation Science*, 17, 351-357, (1983).

[6] Hung, H.K., R.E. Chapman, W.G. Hall, E. Neigut, "A heuristic algorithm for routing and scheduling Dial-A-Ride vehicles", *Presentation at the ORSA/TIMS conference at San Diego*, (1982).

[7] Roy, L. Chapleau, J. Ferland, G. Lapalme, J.M. Rousseau, "The construction of routes and schedules for the transportation of the handicapped", *Working paper, University of Montreal*, (1983).

[8] J. J. Jaw, A. R. Odoni, H. N. Psaraftis and N. H. M. Wilson, "A Heuristic Algorithm for the Multi-Vehicle Advance Request Dial-A-Ride Problem with Time Windows", *Transportation Research* 20B, 243-257, (1986).

[9] Deepak, "Autonomous Dial-A-Ride Transit System: Improvement Method", *Unpublished M. Tech. Thesis, Department IME, I. I. T. Kanpur*, (1996).